

# REASONING-INTENSIVE REGRESSION

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

AI researchers and practitioners increasingly apply large language models (LLMs) to what we call reasoning-intensive regression (RiR), i.e., deducing subtle numerical scores from text. Unlike standard language regression tasks, e.g., for sentiment or similarity, RiR often appears instead in ad-hoc problems such as rubric-based scoring, modeling dense rewards in complex environments, or domain-specific retrieval, where much deeper analysis of context is required while only limited task-specific training data and computation are available. We cast four realistic problems as RiR tasks to establish an initial benchmark, and use that to test our hypothesis that prompting frozen LLMs and finetuning Transformer encoders via gradient descent will both often struggle in RiR. We then propose MENTAT, a simple and lightweight method that combines batch-reflective prompt optimization with neural ensemble learning. MENTAT achieves up to 65% improvement over both baselines, though substantial room remains for future advances in RiR.

## 1 INTRODUCTION

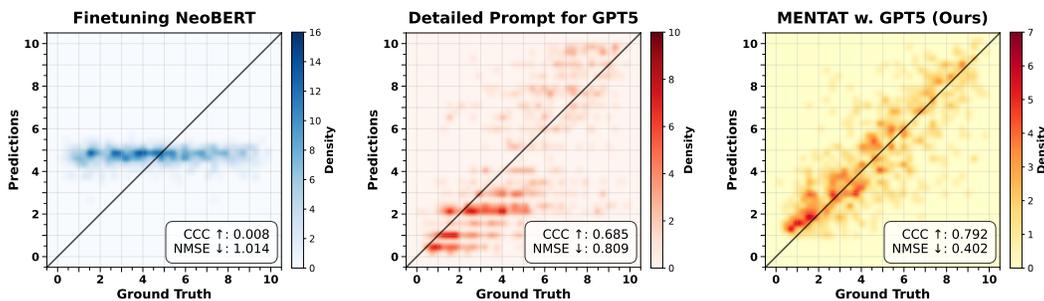


Figure 1: On regression for detecting the first math error, finetuning a NeoBERT model collapses to mean predictions (CCC = 0.01). Meanwhile, detailed (human-crafted) prompting achieves reasonable concordance (CCC = 0.69) but exhibits coarse and imprecise prediction behavior (the dense horizontal lines and near-random NMSE). MENTAT’s performance illustrates how RiR problems benefit from combining deep reasoning capabilities with precise numerical predictions.

Despite fast progress in adapting large language models (LLMs) for building downstream AI systems, lightweight methods for adapting LLMs to even standard *natural-language regression* tasks remain surprisingly elusive (Lukasik et al., 2024b;a; Tang et al., 2024; Song et al., 2025; Song & Bahri, 2025). These tasks, like sentiment analysis, semantic similarity, and document ranking, involve predicting a score  $y \in \mathbb{R}$  from a natural-language string. Surprisingly, on these problems, applying straightforward supervised learning to pretrained Transformer encoders such as BERT Devlin et al. (2019) has been shown to perform competitively with much larger decoder-only LLMs Lukasik et al. (2024a), even with sophisticated fine-tuning methods.

We investigate what we call *Reasoning-Intensive Regression* (RiR), a fuzzy but growing subset of natural-language regression in which processing the text in each instance *demands sequential deduction or deep analysis, rather than shallow identification of features*. Unlike simpler regression tasks, RiR problems call for explicit step-by-step problem decomposition or *reasoning*, where the system produces intermediate sequences of steps like tokens  $\langle r_1, \dots, r_t \rangle \in \Sigma^*$  before committing to a

prediction (Merrill & Sabharwal, 2024). See Figure 2 for a breakdown of regression problems into three levels of complexity: feature-based, semantic analysis, and reasoning-intensive, inspired by Su et al. (2025)’s analysis of retrieval tasks.

These types of applications are emerging rapidly in both research and practice, e.g. to produce scores for ad-hoc applications that process customer calls, student essays, rubric-based LLM generation, or instruction-based query–document relevance (MacDonald, 2024; Es et al., 2024; Su et al., 2025; Thakur et al., 2025). In parallel, the same scoring paradigm is being scaled in recent efforts toward general-purpose chain-of-thought reward models Kimi Team (2025); Ankner et al. (2024a), but these typically assume orders-of-magnitude more labels and compute (e.g., hundreds of thousands of labels in K2) than the lightweight application-specific regimes that are far more common in the long tail.

We establish an initial benchmark for RiR by casting four realistic tasks as regression problems that demand *varying* levels of reasoning: predicting the proportion of a long mathematical deduction up to the first erroneous statement, determining the extent to which an LLM can follow highly composite instructions, predicting the degree to which the response of one Retrieval-Augmented Generation (RAG) system is better than another, and grading student essays on supplied topics. We then identify two practical constraints of downstream applications of RiR: these applications tend to offer only (very) small training sets and have room only for accessible and lightweight computations like LLM inference, lightweight forms of LLM prompt optimization, and finetuning medium-sized neural networks such as small Transformers, but not, say, large-scale reinforcement learning for large language models (DeepSeek-AI, 2025; Kimi Team, 2025).

We ask: *Are there effective methods that are data- and compute-efficient for tackling ad-hoc reasoning-intensive regression problems?* We hypothesize that what makes RiR problems especially challenging is that they combine the reasoning need for *deep analysis of each individual task instance* with the regression challenge of *learning to produce precise, calibrated, and well-ranked scores* from very little data. As illustrated in Figure 1, standard prompt engineering techniques struggle with the high precision needed for learning to approximate a statistical distribution, while approaches that bypass LLM-based reasoning, e.g., training small Transformer encoders, often fail to truly learn RiR problems and instead seek to “hack” the regression loss function by finding degenerate approximations (e.g., collapsing to a small range of scores).

We propose Mistake-Aware prompt Evolver with Neural Training And Testing (MENTAT), a simple and lightweight method that combines iterative prompt optimization with neural regression. Rather than relying on LLMs to produce precise numerical predictions directly, which often results in brittle outputs, MENTAT uses an iterative error-driven prompt evolution process. Starting with even just a very basic prompt, the LLM analyzes its own prediction errors *in large batches*, identifies patterns of its poor performance, and then refines the prompt based on that. After few iterations, MENTAT trains a simple aggregation MLP to reduce multiple rollouts from the LLM-discovered prompt into a final prediction. MENTAT delivers consistent improvements in quality, but nonetheless leaves large headroom on many of the RiR settings we define.

The remainder of the study is as follows: Section 2 describes how we translate four problems into RiR tasks and Section 3 introduces MENTAT. Section 4 presents our evaluation methodology, including the details of our baselines, and the results. The paper concludes with Sections 5 and 6, which synthesize our findings and discuss implications for future research. An extended discussion of related work is given in the Appendix A.

## 2 BENCHMARKING RIR

We collect four tasks for Reasoning-Intensive Regression of varying degrees of reasoning intensity. Refer to Figure 3 for the dataset distributions.

- **Mathematical Error Detection** requires precise logical reasoning and stepwise analysis, while also stressing the fact that LLMs are known to struggle with precisely estimating simple properties like text length.
- **Instruction Following** evaluates how well a response satisfies a set of fine-grained requirements, and expects models to produce calibrated scalar judgments.

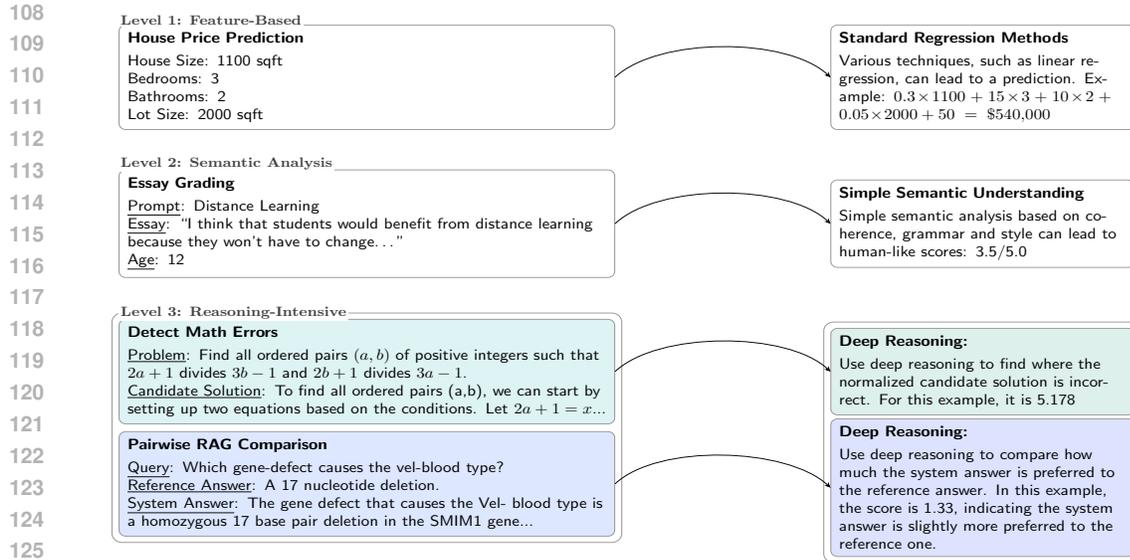


Figure 2: Inspired by Su et al. (2025)’s analysis of retrieval tasks, we break down text-based regression problems into three, informal complexity levels. Level 1 tasks use simple feature-based inputs (for example, the number of bedrooms and bathrooms when predicting home prices). Text-to-text regression achieves strong Level 1 performance with rich datasets (Akhaoui et al., 2025). Level 2 tasks require moderate semantic understanding (sentiment analysis, reward modeling) but are easy for supervised-learning over a pretrained Transformer. Level 3, the focus of this work, represents Reasoning-Intensive Regression (RiR), which requires deep sequential reasoning.

- **Pairwise RAG Comparison** emphasizes the need for models to perform nuanced human-like judgment and contextual understanding, demanding sophisticated evaluative reasoning.
- **Essay Grading** serves as a reference point, requiring semantic understanding where encoders like BERT might already perform well with a reasonable amount of finetuning data.

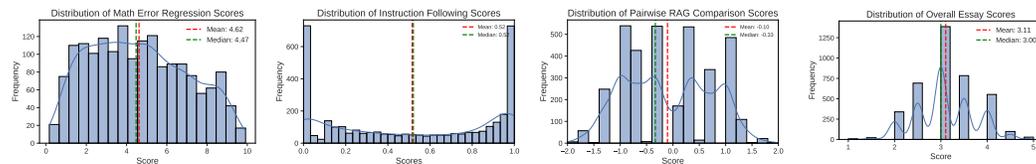


Figure 3: Ground-truth score distributions for mathematical error detection (the spread capturing the tendency for solutions to fail towards the center), instruction following (capturing the tendency to favor the tails), pairwise RAG comparison (narrow distribution around averaged judgments), and essay grading (tight clustering characteristic of qualitative assessments).

**Regression Metrics** Normalized Mean Square Error (NMSE) is a common metric for reporting regression performance:  $\sum_i^n (y_i - \hat{y}_i)^2 / \sum_i^n (y_i - \bar{y})^2$ , where  $n$  is the size of the dataset,  $\hat{y}_i$  is a prediction,  $y_i$  the corresponding ground truth value, and  $\bar{y}$  is the mean of the set.

But distance-based metrics are inadequate for typical RiR problems; RiR systems can artificially lower their NMSE simply by avoiding “risky” predictions at the extremes. This can be seen in Figure 1 earlier, particularly in comparing the fine-tuned NeoBERT model Breton et al. (2025) against detailed (human-crafted) prompting. Following Figure 1, if we were to rely on NMSE, detailed prompting for GPT-5 would not appear to substantially outperform NeoBERT (0.81 vs. 1.01), and this gap would be even reversed for weaker LLMs. Examining the distribution of predictions reveals

that NeoBERT “hacked” the loss function by learning a collapsed distribution, while the prompted LLM actually shows substantial signs of ranking the inputs correctly.

This can be captured in a Concordance Correlation Coefficient (CCC) of 0.01 for NeoBERT versus a CCC of 0.69 for detailed prompting. We thus suggest the use of the CCC as an additional, and perhaps more appropriate, RiR metric. CCC measures both correlation and agreement, defined as  $\frac{2\rho\sigma_y\sigma_{\hat{y}}}{\sigma_y^2+\sigma_{\hat{y}}^2+(\mu_y-\mu_{\hat{y}})^2}$ , where  $\rho$  is the Pearson correlation coefficient between predictions  $\hat{y}$  and ground truth  $y$ ,  $\sigma_y$  and  $\sigma_{\hat{y}}$  are their respective standard deviations, and  $\mu_y$  and  $\mu_{\hat{y}}$  are their means. CCC penalizes systematic bias and rewards predictions that maintain the natural variance of the distribution.

**Detecting Mathematical Errors** We derive a dataset for predicting the *fraction* of a mathematical solution up to the first erroneous reasoning step, given a problem and incorrect solution in LaTeX, from ProcessBench (Zheng et al., 2024). To effectively do this, a model must systematically reason formally about math steps rather than relying on probabilistic heuristics, but it must also be good at estimating relative lengths and inferring the boundaries of the steps in a calibrated way.

To convert the original classification task into a regression problem, we first filter out problems with correct solutions or final answers. We then merge all solution steps into a single continuous text  $T = s_1\|s_2\|\dots\|s_n$  (here  $\|$  denotes concatenation). Next, for a solution with error at step  $k$ , the regression score  $R$  is  $10 \times (\sum_{i=1}^{k-1} |s_i| + \frac{1}{2}|s_k|)/|T|$  where  $|s_i|$  denotes the length of step  $i$ , and  $|T|$  is the total length of the concatenated solution. See an example entry in Appendix E.

**Instruction Following** We derive a task from the WILDIFEVA corpus Lior et al. (2025) that targets *instruction-following* in long-form generation. Each example consists of: (i) a user *task* prompt; (ii) a list of atomic requirements (the *decomposition*); (iii) a *model answer* produced by Llama-3.1-8B (zero-shot); and (iv) per-requirement satisfaction scores originally produced by Llama-3.1-70B acting as an automatic judge. The goal is to predict a single continuous label  $y \in [0, 1]$  that reflects the overall degree to which the answer adheres to the decomposed instructions. More precisely, for each decomposition instance, the judge produced a probability-like score  $s_i \in [0, 1]$  for each requirement  $r_i, i = 1, \dots, K$ . We then use the harmonic mean of these scores as our overall judgment, emphasizing the need to adhere well to all task requirements. To test instruction following, we do not expose the decomposition to NeoBERT or an LLM; instead, they are only given the task and model answer and must infer the overall score.

**Pairwise RAG Comparison** We derive a dataset for comparing two LLM outputs on a scale from the RAG-QA evaluations (Han et al., 2024). Each query  $q \in \mathcal{Q}$  has responses  $A_1, A_2$  and a target comparative score from  $-2$  to  $2$  representing the *average* annotation of three human judges, who were instructed to assess response helpfulness, truthfulness, and completeness. Here, positive scores means that  $A_1$  is better (and vice versa). This task partially aligns with RiR as judging the outputs and comparing them in light of each query often requires nuanced judgment.

**Essay Grading** We lastly use an essay grading dataset Crossley et al. (2023), where each entry contains among other features an essay prompt, a student (grade 8–12) response, associated demographic information, and an overall score between 1 and 5. Although Essay Grading is simpler than the rest, it serves as a reference point for the other RiR tasks.

We evaluate these tasks using two proprietary LLMs (GPT-4.1, GPT-5) across three tasks, plus an open-source model (gpt-oss-20b) on Instruction Following for reproducibility and generalization validation.

### 3 MENTAT

MENTAT combines two simple ideas, depicted in Figure 4: it allows the LLM itself to reflect in batches to incrementally adjust its own prompt, and it aggregates multiple rollouts from the optimized LLM system with a simple trained MLP.

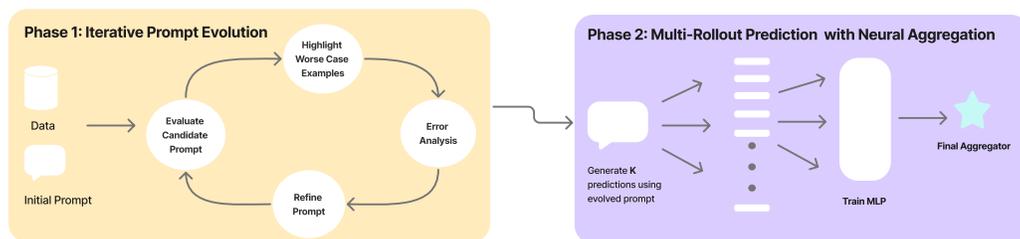


Figure 4: Phase 1 performs prompt evolution through iterative and *batched* reflection. Given a candidate prompt, we collect rollouts on  $n$  samples, divided into training and validation sets. A model instructed to focus on the  $\sqrt{n}$  worst-performing examples (selected based on absolute prediction error) analyzes the rollouts on the training samples, in light of the optimization history from previous iterations, and makes proposals that refine the prompt. This cycle continues for a predetermined number of iterations, after which we select the best-performing prompt  $P_{\text{best}}$  as evaluated on the validation set, where by best we mean the prompt that led to the highest CCC value. Phase 2 generates multi-rollout predictions by applying  $P_{\text{best}}$  and aggregating  $K$  stochastic predictions per input and trains a neural aggregator  $f_{\theta}$  on sorted rollouts using a combined CCC–NMSE loss. Test predictions are obtained by sampling test rollouts and applying the trained aggregator  $f_{\theta}$ .

**Phase 1: Prompt Evolution** MENTAT’s first step is to make sure that the LLM prompt reflects both *local* instructions for reasoning about each input and *global* guidance about the distribution of ground-truth scores. Though any approach for prompt optimization can be used here, e.g., MIPRO Opsahl-Ong et al. (2024) or GEPA Agrawal et al. (2025), through preliminary experiments we identified two special properties in RiR tasks that call for different design choices.

First, performing rollouts with powerful *reasoning* models can be expensive and slow, when compared to standard LLMs, for which existing optimizers were built. To remain within the lightweight constraints of typical RiR tasks, a suitable prompt evolution stage would have to minimize *both* the number of rollouts performed with the LLM and the number of *inherently sequential stages* or iterations of optimization. **Second, RiR tasks require attention to distributional properties, calibration, variance matching, and avoiding collapse to mean predictions, beyond per-example accuracy. This is because MENTAT’s aggregation design demonstrates that it can be easy to turn a well-calibrated system into one that has low pointwise error, but the reverse is not necessarily true.**

This motivates us to test an exceedingly simple reasoning-based technique for optimizing LLM systems that contain a single prompt.<sup>1</sup> While batch-based prompt optimization has been extensively explored in prior work Pryzant et al. (2023); Ye et al. (2024), we focus on combining it with neural aggregation specifically for regression tasks, using CCC alongside NMSE to guide prompt selection and aggregator training. This simple design is inspired by human prompt engineering practice (Husain & Shankar, 2024).

Concretely, we proceed in a very small number of sequential iterations (three in our experiments). In each iteration, the work is highly parallelizable: we evaluate the current prompt on a shuffled sample of the training set, and then concatenate all of the rollouts for analysis by the same LLM. It is then asked to identify systematic errors by analyzing the worst-performing examples and to generate improved instructions. In each iteration, the LLM receives three key inputs: current instructions, performance analysis with detailed error patterns, and a formatted history of previous optimization attempts. This historical context prevents the method from cycling through previously unsuccessful approaches and enables progressive refinement. At the end of this process, the best-performing prompt (via NMSE or CCC) is selected on a separate validation set.

In our evaluation, to stress MENTAT, we start from a deliberately basic prompt for each task, to reflect a more challenging setting.<sup>2</sup> Note also that this iterative prompt evolution follows a single optimization

<sup>1</sup>We leave extending this method to multi-stage LLM programs and conducting an extensive comparison of different prompt optimization strategies to future work.

<sup>2</sup>Examples of the basic vs. the detailed prompts used for the four tasks can be found in Appendices H and F), respectively. They differ in the inclusion of detailed procedural steps, calibration guidance, and/or domain-specific heuristics that human experts may decide to include.

trajectory. In principle, MENTAT could employ multiple random restarts, which could be parallelized to explore diverse regions of the prompt space. However, we focus on single-trajectory optimization both for computational efficiency and algorithmic simplicity.

**Phase 2: Multi-Rollout Generation with Neural Aggregation** Using the best LLM-discovered prompt from Phase 1, MENTAT generates multiple independent predictions for each example. The multi-rollout approach captures the inherent uncertainty in LLM predictions, as each rollout can reason independently, and provides richer signal for the subsequent neural aggregation phase. In practice, we set this to three rollouts per example.

We train a small Multi-Layer Perceptron (MLP) to combine rollout predictions. The aggregator ensures order invariance by sorting rollout predictions, incorporates statistical features (mean, standard deviation, min, max), and is optimized for a combination of the CCC and NMSE loss functions. Overall, this method builds on self-consistency Wang et al. (2023) and best-of-N voting Stinson et al. (2020); Snell et al. (2024), but differs by training a lightweight aggregator that learns task-specific weighting of rollout statistics rather than using fixed aggregation rules.

**Analysis of MENTAT** Figure 5 shows rollout variance distributions for detailed (human-crafted) versus MENTAT-evolved prompts across three tasks. MENTAT’s prompt evolution consistently reduces variance on reasoning-intensive tasks, achieving a 30% reduction in mean variance on Mathematical Error Detection. This demonstrates that evolved prompts produce more stable reasoning patterns rather than merely providing noisy signals for the aggregator to smooth. However, non-trivial variance remains after evolution, enabling the neural aggregator to extract meaningful signal from rollout diversity. These findings reveal MENTAT’s complementary design: prompt evolution improves prediction reliability while neural aggregation refines these consistent signals into precise numerical outputs.

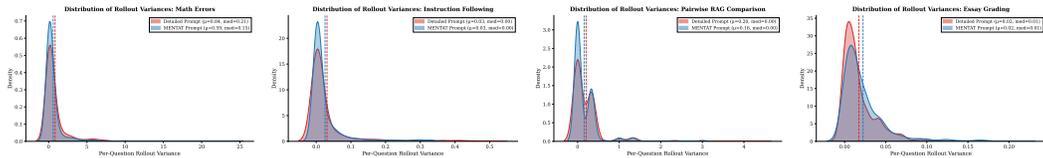


Figure 5: Distribution of per-question rollout variances comparing the Detailed (human-crafted) prompt against the MENTAT-evolved prompt across three tasks. For reasoning-intensive tasks (Mathematical Error Detection and Pairwise RAG Comparison), MENTAT’s prompt evolution yields lower mean rollout variance, indicating more consistent predictions across independent rollouts. In contrast, Essay Grading, which is characterized as a Level 2 (semantic analysis) task requiring less sequential reasoning, shows comparable variance between prompts. This pattern suggests that prompt evolution yields the greatest consistency gains on tasks where deep reasoning is essential, while contributing less when shallow semantic features suffice.

## 4 EVALUATION

We define two standard baselines for RiR problems: fine-tuning a small Transformer encoder and prompting an LLM, and use these two to understand the relative merits of our method MENTAT and to develop a series of ablation experiments. Additionally, we compare against Agrawal et al. (2025), a recent reflective prompt optimization method, to situate MENTAT relative to modern prompt optimizers.

### 4.1 BASELINE: FINETUNING A TRANSFORMER ENCODER

We formulate RiR as supervised regression using a 250M-parameter NeoBERT model. The architecture processes minimally formatted text sequences (e.g., combining problem statements with solutions for math errors, augmented with domain-specific prompts).

Inputs are tokenized using NeoBERT’s byte-level BPE tokenizer, truncated or padded to 1024 tokens, and passed through the pretrained encoder. The model extracts representations from the [CLS] token, applies dropout regularization ( $p = 0.2$ ), and uses linear projection for scalar predictions.

324 The optimization objective minimizes weighted NMSE and CCC using AdamW (Loshchilov &  
325 Hutter, 2019). This architecture requires only prompt templating beyond standard fine-tuning, with  
326 hyperparameters detailed in Appendix D.1.

## 328 4.2 BASELINE: PROMPTING A LARGE LANGUAGE MODEL

329  
330 We employ Chain-of-Thought style prompting to encourage frozen LLMs to perform explicit rea-  
331 soning through step-by-step token generation. Our evaluation uses two proprietary models with  
332 different reasoning capabilities: GPT-4.1 (non-reasoning) and GPT-5 (reasoning) across three tasks  
333 (Mathematical Error Detection, Pairwise RAG Comparison, and Essay Grading). For the Instruction  
334 Following task, we employ gpt-oss-20b, an open-source model, to demonstrate that our method  
335 generalizes beyond proprietary systems and to provide more easily reproducible baselines for the  
336 community. The detailed prompts for all tasks help guide the decomposition of complex inputs and  
337 the templates can be found in Appendix F.

338 This approach is motivated by several practical advantages. Frozen LLMs can act as a unified  
339 interface across various natural language tasks, with no or very little training data. This is especially  
340 valuable in reasoning-intensive regression (RiR) tasks where annotated datasets are often scarce.  
341 Utilizing a shared, unified, and amortized infrastructure (i.e., LLM servers) enables us to deploy a  
342 single model across many tasks, significantly reducing the computational and financial overhead  
343 compared to training multiple specifiable models.

## 344 4.3 EXPERIMENTAL SETUP

345  
346 Our experimental design evaluates MENTAT across four reasoning-intensive regression tasks using  
347 a structured approach. We test GPT-4.1 and GPT-5 on three tasks (Mathematical Error Detection,  
348 Pairwise RAG Comparison, and Essay Grading), while Instruction Following uses gpt-oss-20b to  
349 demonstrate generalization to open-source models.

350 For the three tasks using proprietary models, we employ 750 test examples with results averaged  
351 across three independent runs. We evaluate under two training configurations (100 and 500 samples)  
352 that reflect real-world data constraints typical in ad-hoc RiR applications. **For prompt optimization  
353 methods (including MENTAT), we use balanced train/validation splits of 50+50 and 250+250 samples;  
354 Phase 1 uses these for prompt evolution, and Phase 2 generates 3 rollouts per training sample for  
355 MLP training. For NeoBERT finetuning, we employ training-heavy splits of 50 + 50 and 350 + 150  
356 samples to leverage the model’s supervised learning capabilities.**

357 For Instruction Following, we use a single configuration with 500 training, 500 validation, and  
358 2000 test samples, reflecting the different data availability typical for this fundamental capability  
359 assessment. This larger test set enables more robust evaluation of the nuanced instruction-adherence  
360 requirements.

361 This experimental structure allows us to assess MENTAT’s effectiveness across different model capa-  
362 bilities (reasoning vs. non-reasoning), data regimes (limited vs. moderate training data), and model  
363 accessibility (proprietary vs. open-source), providing comprehensive validation of our approach for  
364 practical RiR applications.

### 366 4.3.1 COMPUTATION COST

367  
368 **MENTAT’s computational costs comprise two phases. At inference time, each prediction requires 3**  
369 **rollouts, resulting in  $3\times$  token cost compared to single-pass prompting. However, all rollouts can be**  
370 **generated in parallel, so wall-clock latency remains approximately equivalent to a single rollout in**  
371 **parallelized deployment scenarios.**

372 **During optimization (Phase 1), MENTAT uses a fixed 3-iteration design. Each iteration evaluates the**  
373 **current prompt on all  $n = 250$  samples (parallelizable) and performs one reflection call analyzing the**  
374  **$\sqrt{250} \approx 16$  worst-performing examples, totaling approximately 753 LLM calls across 3 sequential**  
375 **steps. GEPA’s “light” configuration converges after an average of 23 sequential iterations (ranging 15-**  
376 **34 across runs). GEPA’s evolutionary search thus requires approximately  $8\times$  more sequential rounds**  
377 **than MENTAT’s fixed design, providing MENTAT a substantial wall-clock advantage in parallelized**  
**deployments, though total token consumption may differ.**

LM	Method	Math Errors				Pairwise RAG				Essay Grading			
		NMSE ↓		CCC ↑		NMSE ↓		CCC ↑		NMSE ↓		CCC ↑	
		100	500	100	500	100	500	100	500	100	500	100	500
<b>Main Methods</b>													
NeoBERT	Gradient Descent	1.05	1.01	0.02	0.06	1.44	1.02	0.02	0.10	1.03	0.91	0.19	0.65
GPT-4.1	Basic Prompt	1.59	1.59	0.36	0.36	2.18	2.18	0.47	0.47	0.75	0.75	0.63	0.63
	Detailed Prompt	1.13	1.13	0.52	0.52	2.20	2.20	0.47	0.47	0.73	0.73	0.65	0.65
	MENTAT <sub>Basic Prompt</sub>	0.87	0.76	0.51	0.49	<b>0.77</b>	<b>0.80</b>	<b>0.50</b>	<b>0.52</b>	<b>0.54</b>	<b>0.53</b>	<b>0.70</b>	<b>0.68</b>
GPT-5	Basic Prompt	0.77	0.77	0.66	0.66	2.25	2.25	0.35	0.35	1.31	1.31	0.42	0.42
	Detailed Prompt	0.78	0.78	0.69	0.69	2.18	2.18	0.31	0.31	1.53	1.53	0.40	0.40
	MENTAT <sub>Basic Prompt</sub>	<b>0.52</b>	<b>0.42</b>	<b>0.72</b>	<b>0.78</b>	1.07	0.93	0.36	0.33	0.64	0.67	0.59	0.55
<b>Ablations</b>													
GPT-4.1	MENTAT Prompt	1.39	1.29	0.45	0.48	2.00	1.69	0.45	0.48	0.61	0.71	0.68	0.66
	MENTAT-Avg	1.00	1.01	0.52	0.52	1.82	1.48	0.48	0.51	0.57	0.63	0.69	<b>0.68</b>
	GEPA	1.04	1.01	0.49	0.54	2.16	2.40	0.44	0.43	0.79	0.81	0.63	0.63
GPT-5	MENTAT Prompt	0.66	0.58	0.66	0.72	1.43	1.95	0.33	0.30	0.74	0.70	0.57	0.54
	MENTAT-Avg	0.59	0.51	0.68	0.75	1.31	1.83	0.35	0.32	0.69	0.67	0.57	0.55
	GEPA	0.78	0.63	0.68	0.69	2.48	2.29	0.28	0.28	1.01	1.01	0.42	0.44

Table 1: Performance comparison across Mathematical Error Detection, Pairwise RAG Comparison, and Essay Grading using GPT-4.1 and GPT-5 as our models. Each entry is the average of three independent runs on a test set of size 750. Total training sizes are 100 and 500 (train/val combined). **Ablations:** MENTAT Prompt uses only error-driven prompt refinement on training data *starting from a basic prompt*; MENTAT-Avg shows performance when replacing the trained MLP with averaging. We remark here that NeoBERT obtains an average NMSE and CCC of 0.60 and 0.66 respectively, on a training regime of 1500 (1000 training + 500 validation) on Pairwise RAG Comparison. That is, NeoBERT needs much more data on this task to lead to good performance, but it can be achieved. **This table along with additional reporting of standard deviation can be found in Appendix 3.**

LM	Method	Instruction Following	
		NMSE ↓	CCC ↑
NeoBERT	Gradient Descent	1.08 (0.07)	0.36 (0.04)
GPT-OSS-20B	Basic Prompt	1.18 (0.00)	0.32 (0.00)
	Detailed Prompt	1.16 (0.00)	0.33 (0.00)
	MENTAT <sub>Basic Prompt</sub>	0.95 (0.09)	0.42 (0.01)
	MENTAT <sub>Detailed Prompt</sub>	<b>0.90 (0.04)</b>	0.43 (0.00)
<b>Ablations</b>			
GPT-OSS-20B	MENTAT <sub>Basic Prompt</sub> Prompt	1.25 (0.05)	0.35 (0.01)
	MENTAT <sub>Basic Prompt</sub> -Avg	1.06 (0.04)	0.38 (0.02)
	MENTAT <sub>Detailed Prompt</sub> Prompt	1.24 (0.13)	0.36 (0.01)
	MENTAT <sub>Detailed Prompt</sub> -Avg	1.09 (0.06)	0.39 (0.02)
	GEPA	1.06 (0.02)	<b>0.46 (0.01)</b>

Table 2: Performance on the Instruction Following task using the gpt-oss-20b model. Each entry is the average of three independent runs on a test set of size 2000. Total training configuration uses 500 training and 500 validation samples. Ablations: MENTAT Prompt uses only error-driven prompt refinement on training data; MENTAT-Avg shows performance when replacing the trained MLP with averaging. The subscripts, basic prompt and detailed prompt, are what we use as the initial prompt in the MENTAT framework. **Moreover, values within the parenthesis represent standard deviations.**

Phase 2 (MLP training) has negligible cost, as the MLP contains only 8 hidden units and trains on 750 rollout vectors (250 samples  $\times$  3 rollouts each).

#### 4.4 RESULTS

Our main evaluation results are reported in Table 1 and 2, demonstrating significant performance variations across methods and tasks. The results reveal distinct patterns in how different approaches handle reasoning-intensive regression problems, with MENTAT consistently outperforming baseline methods across most configurations. **Beyond aggregate metrics, we analyze failure modes across methods: NeoBERT’s distribution collapse (1), GPT-5’s center-seeking behavior on pairwise RAG**

(6), and systematic quantization patterns in LLM outputs (Appendix B). We provide additional per-task qualitative error analysis in Appendix B.

**Mathematical Error Detection Performance** On this task, finetuning NeoBERT achieves near-zero CCC scores across both training configurations and effectively collapsing to mean predictions as shown in Figure 1. In contrast, LLM-based approaches demonstrated substantial reasoning capabilities. GPT-4.1 with detailed prompting achieved CCC scores of 0.52 (100-sample training) and maintained this performance at 500 samples. However, MENTAT with GPT-4.1 showed only modest improvements, reaching CCC scores of 0.51 (100 samples) and 0.49 (500 samples), representing approximately stable performance with slight variation. We hypothesize that GPT-4.1’s limited reasoning capabilities on this reasoning-intensive task made it difficult to understand its own errors and thus improve.

The most dramatic improvements can be seen with GPT-5. While detailed prompting with GPT-5 achieved strong baseline performance (CCC: 0.69, NMSE: 0.78), MENTAT with GPT-5 delivered substantial enhancements. In the 100-sample training regime, CCC improved by 4.3%, while NMSE improved by 33.3%. In the 500-sample training regime, CCC improved by 13%, while NMSE improved by 46.2%. These results indicate that MENTAT’s iterative prompt refinement and neural aggregation effectively leverage GPT-5’s reasoning capabilities while addressing the precision limitations inherent in direct LLM numerical prediction.

**Instruction Following Performance** On instruction following, NeoBERT achieved modest performance (CCC: 0.36, NMSE: 1.08), while `gpt-oss-20b` with basic and detailed prompting showed similar limitations (CCC: 0.32-0.33, NMSE: 1.16-1.18). Both approaches struggled with the nuanced evaluation required for assessing instruction adherence.

MENTAT demonstrated clear improvements across both initialization strategies. Starting from the basic prompt, MENTAT achieved CCC of 0.42 and NMSE of 0.95 (31% and 19% improvements respectively). With detailed prompt initialization, MENTAT reached CCC of 0.43 and NMSE of 0.90 (30% and 22% improvements). The ablation results show that both prompt evolution and neural aggregation contribute to MENTAT’s success on this task, with the full method consistently outperforming the individual components across both initialization strategies.

**Pairwise RAG Comparison Performance** On the pairwise RAG comparison task, finetuning NeoBERT achieved very low CCC scores while appearing competitive on the NMSE metric by “hacking” the distribution. Surprisingly, GPT-4.1 demonstrated superior performance compared to GPT-5 on this task, in sharp contrast with the general trend observed in mathematical error detection. Detailed prompting with GPT-4.1 achieved CCC scores of 0.47 across both training configurations, while GPT-5 detailed prompting resulted in lower CCC scores of 0.31.

Unlike math errors, instruction following, and essay grading tasks, correct decisions on the pairwise RAG benchmark often hinge on a few salient cues and short justifications. With chain-of-thought scaffolds on this task, we observe that GPT-5 systematically “overthinks,” resulting in predictions that concentrate near the center (0 on the  $[-2, 2]$  margin) rather than faithfully spreading across the empirical label distribution. As shown in Figure 6, its variance is under-dispersed relative to ground truth, with more than half of examples yielding identical rollouts across three samples. Rollout correlations are very high, and the final numbers fall on a coarse grid (e.g.,  $\{-1, -\frac{1}{3}, 0, \frac{1}{3}\}$ ), all consistent with hedging.

By contrast, GPT-4.1 produces short, decisive judgments that remain closer to the dataset mean with greater spread and more frequent use of the extremes. Although GPT-4.1 rollouts are also correlated, the resulting distribution retains enough variance and calibration to yield substantially higher CCC. For pairwise RAG, GPT-5 tends toward the center and compresses its numeric range, degrading distributional fidelity (and thus CCC) even when NMSE remains similar.

We hypothesize that GPT-4.1’s superior performance on pairwise RAG comparison aligns with recent findings that large reasoning models often underperform on simpler tasks (Shojaee et al., 2025). These models initially find correct solutions but continue reasoning toward incorrect answers, suggesting that excessive sophisticated reasoning can sometimes be counterproductive. This hypothesis is supported by our observation that over half of GPT-5’s examples yield identical rollouts across three samples, with final scores clustering on a coarse grid rather than reflecting the task’s inherent variance.

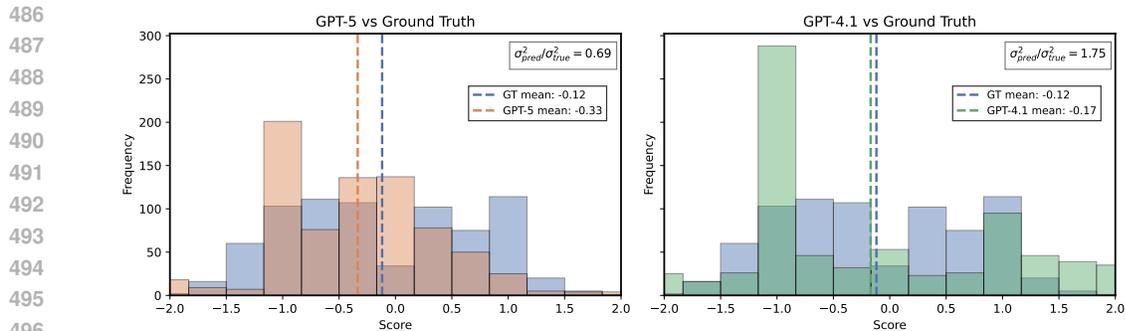


Figure 6: Pairwise RAG distributions on the mean of three rollouts vs. ground truth after the prompt evolution process. GPT-5 (left) is center-seeking and under-dispersed; GPT-4.1 (right) stays closer to the empirical mean and exhibits greater spread. This behavior leads to higher CCC for GPT-4.1.

**Essay Grading Performance** Essay grading represented the least complex reasoning-intensive task, with NeoBERT achieving reasonable performance that improved substantially with additional training data. This aligns with the task’s characterization as requiring primarily semantic understanding rather than deep sequential reasoning. GPT-4.1 achieved strong baseline performance with detailed prompting (CCC: 0.65, NMSE: 0.73), while MENTAT provided meaningful improvements. In the 100-sample training regime, CCC improved by 7.7% and NMSE improved by 26.0% compared to detailed prompting. In the 500-sample training regime, CCC improved by 4.6% and NMSE improved by 27.4%. Notably, GPT-5 performance on essay grading showed surprisingly poor concordance compared to GPT-4.1, supporting the hypothesis discussed in Section 4.4 that sophisticated reasoning models may over-deliberate on simpler tasks.

## 5 CONCLUSION

We investigated *reasoning-intensive regression* (RiR). Our empirical findings reveal tension: prompting leverages LLMs’ reasoning capabilities but produces quantized, imprecise outputs, while supervised finetuning for regression can often collapse without learning the task. We proposed MENTAT, a simple method that suggests that hybrid approaches may help address this tension through iteratively optimizing the prompts via batched error analysis combined with neural aggregation, achieving consistent improvements across RiR tasks.

However, our work opens several rich avenues for future research. The RiR framework we establish creates opportunities to systematically evaluate sophisticated prompt optimization techniques such as GEPA and develop RiR-adapted regression-aware finetuning methods (Lukasik et al., 2024b; Chiang et al., 2025). Our lightweight constraint focus also motivates exploring the efficiency-performance trade-offs in reasoning-intensive tasks. While reinforcement learning methods like Group Relative Policy Optimization Shao et al. (2024) require thousands of rollouts beyond current practical limits, our benchmark provides a testbed for developing more efficient alternatives as RiR datasets scale. Similarly, MENTAT’s  $3\times$  inference cost increase highlights the need for systematic cost-benefit analysis across deployment scenarios, opening questions about adaptive rollout strategies and inference-time optimization that our tasks can help address.

## REFERENCES

- Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziem, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gega: Reflective prompt evolution can outperform reinforcement learning, 2025. URL <https://arxiv.org/abs/2507.19457>.
- Yash Akhauri, Bryan Lewandowski, Cheng-Hsi Lin, Adrian N. Reyes, Grant C. Forbes, Arissa Wongpanich, Bangding Yang, Mohamed S. Abdelfattah, Sagi Perel, and Xingyou Song. Performance

- 540 prediction for large systems via text-to-text regression, 2025. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2506.21718)  
541 [2506.21718](https://arxiv.org/abs/2506.21718).
- 542
- 543 James Urquhart Allingham, Jie Ren, Michael W. Dusenberry, Xiuye Gu, Yin Cui, Dustin Tran,  
544 Jeremiah Zhe Liu, and Balaji Lakshminarayanan. A simple zero-shot prompt weighting technique  
545 to improve prompt ensembling in text-image models. In *Proceedings of the 40th International*  
546 *Conference on Machine Learning*, ICML'23. JMLR.org, 2023.
- 547 Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan D Chang, and Prithviraj Ammanabrolu.  
548 Critique-out-loud reward models. *arXiv preprint arXiv:2408.11791*, 2024a.
- 549
- 550 Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan D. Chang, and Prithviraj Ammanabrolu.  
551 Critique-out-loud reward models, 2024b. URL <https://arxiv.org/abs/2408.11791>.
- 552
- 553 Simran Arora, Avanika Narayan, Mayee F. Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami,  
554 Frederic Sala, and Christopher R. Ask me anything: A simple strategy for prompting language  
555 models, 2022. URL <https://arxiv.org/abs/2210.02441>.
- 556 Daniel Beaglehole, Adityanarayanan Radhakrishnan, Enric Boix-Adser, and Mikhail Belkin. Ag-  
557 gregate and conquer: detecting and steering llm concepts by combining nonlinear predictors over  
558 multiple layers, 2025. URL <https://arxiv.org/abs/2502.03708>.
- 559 L. Breiman. Stacked regressions. *Machine Learning*, 24:49–64, 1996a. URL [https://api.](https://api.semanticscholar.org/CorpusID:27026167)  
560 [semanticscholar.org/CorpusID:27026167](https://api.semanticscholar.org/CorpusID:27026167).
- 561
- 562 Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123140, August 1996b. ISSN 0885-6125.  
563 doi: 10.1023/A:1018054314350. URL <https://doi.org/10.1023/A:1018054314350>.
- 564
- 565 Leo Breiman. Random forests. *Mach. Learn.*, 45(1):532, October 2001. ISSN 0885-6125. doi:  
566 10.1023/A:1010933404324. URL <https://doi.org/10.1023/A:1010933404324>.
- 567
- 568 Lola Le Breton, Quentin Fournier, Mariam El Mezouar, John X. Morris, and Sarath Chandar. Neobert:  
569 A next-generation bert, 2025. URL <https://arxiv.org/abs/2502.19587>.
- 570
- 571 Cheng-Han Chiang, Hung yi Lee, and Michal Lukasik. Tract: Regression-aware fine-tuning  
572 meets chain-of-thought reasoning for llm-as-a-judge, 2025. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2503.04381)  
[2503.04381](https://arxiv.org/abs/2503.04381).
- 573
- 574 Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam  
575 Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh,  
576 Kensen Shi, Sashank Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay,  
577 Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope,  
578 James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm  
579 Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra,  
580 Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret  
581 Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick,  
582 Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica  
583 Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan  
584 Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck,  
585 Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: scaling language modeling with pathways. *J.*  
*Mach. Learn. Res.*, 24(1), March 2024. ISSN 1532-4435.
- 586
- 587 Scott A. Crossley, Yu Tian, Perpetual Baffour, Alex Franklin, Youngmeen Kim, Wesley Morris, Meg  
588 Benner, Aigner Picou, and Ulrich Boser. The english language learner insight, proficiency and  
589 skills evaluation (ellipse) corpus. *International Journal of Learner Corpus Research*, 2023. URL  
<https://api.semanticscholar.org/CorpusID:267599728>.
- 590
- 591 Rbert Csords, Piotr Pikos, Kazuki Irie, and Jrgen Schmidhuber. Switchhead: Accelerating transform-  
592 ers with mixture-of-experts attention, 2024. URL <https://arxiv.org/abs/2312.07987>.
- 593
- 594 DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning,  
2025. URL <https://arxiv.org/abs/2501.12948>.

- 594 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep  
595 bidirectional transformers for language understanding, 2019. URL [https://arxiv.org/abs/  
596 1810.04805](https://arxiv.org/abs/1810.04805).
- 597  
598 Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. Ragas: Automated evaluation  
599 of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter  
600 of the Association for Computational Linguistics: System Demonstrations*, pp. 150–158, 2024.
- 601 William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: scaling to trillion parameter  
602 models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23(1), January 2022. ISSN  
603 1532-4435.
- 604  
605 Patrick Fernandes, Daniel Deutsch, Mara Finkelstein, Parker Riley, André Martins, Graham Neubig,  
606 Ankush Garg, Jonathan Clark, Markus Freitag, and Orhan Firat. The devil is in the errors: Leverag-  
607 ing large language models for fine-grained machine translation evaluation. In Philipp Koehn, Barry  
608 Haddow, Tom Kocmi, and Christof Monz (eds.), *Proceedings of the Eighth Conference on Machine  
609 Translation*, pp. 1066–1083, Singapore, December 2023. Association for Computational Linguis-  
610 tics. doi: 10.18653/v1/2023.wmt-1.100. URL <https://aclanthology.org/2023.wmt-1.100/>.
- 611 Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings  
612 of the Thirteenth International Conference on International Conference on Machine Learning,  
613 ICML’96*, pp. 148156, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc. ISBN  
614 1558604197.
- 615 M.A. Ganaie, Minghui Hu, A.K. Malik, M. Tanveer, and P.N. Suganthan. Ensemble deep learning: A  
616 review. *Engineering Applications of Artificial Intelligence*, 115:105151, 2022. ISSN 0952-1976.  
617 doi: <https://doi.org/10.1016/j.engappai.2022.105151>. URL [https://www.sciencedirect.com/  
618 science/article/pii/S095219762200269X](https://www.sciencedirect.com/science/article/pii/S095219762200269X).
- 619 Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are  
620 zero-shot time series forecasters, 2024. URL <https://arxiv.org/abs/2310.07820>.
- 621  
622 Rujun Han, Yuhao Zhang, Peng Qi, Yumo Xu, Jenyuan Wang, Lan Liu, William Yang Wang, Bonan  
623 Min, and Vittorio Castelli. Rag-qa arena: Evaluating domain robustness for long-form retrieval  
624 augmented question answering. 2024. URL <https://arxiv.org/abs/2407.13998>.
- 625  
626 Yi-Chong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Bing Qin, and Ting Liu.  
627 Ensemble learning for heterogeneous large language models with deep parallel collaboration. 2024.  
628 URL <https://api.semanticscholar.org/CorpusID:269282634>.
- 629 Hamel Husain and Shreya Shankar. Ai evals for engineers & pms. [https://maven.com/  
630 parlance-labs/evals](https://maven.com/parlance-labs/evals), 2024. Cohort-based course on building and iterating evaluation sys-  
631 tems for AI products.
- 632  
633 Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures  
634 of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.
- 635  
636 Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models  
637 with pairwise ranking and generative fusion, 2023. URL <https://arxiv.org/abs/2306.02561>.
- 638  
639 Mingjian Jiang, Yangjun Ruan, Sicong Huang, Saifei Liao, Silviu Pitis, Roger Baker Grosse, and  
640 Jimmy Ba. Calibrating language models via augmented prompt ensembles. URL [https://api.  
641 semanticscholar.org/CorpusID:271797871](https://api.semanticscholar.org/CorpusID:271797871).
- 642  
643 M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the em algorithm. In *Proceedings  
644 of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 2, pp.  
645 1339–1344 vol.2, 1993. doi: 10.1109/IJCNN.1993.716791.
- 646  
647 Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. Exploring  
648 demonstration ensembling for in-context learning. In *ICLR 2023 Workshop on Mathematical and  
649 Empirical Understanding of Foundation Models*, 2023.
- 650  
651 Kimi Team. Kimi K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

- 648 Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen.  
649 Making large language models better reasoners with step-aware verifier, 2023. URL <https://arxiv.org/abs/2206.02336>.  
650
- 651 Gili Lior, Asaf Yehudai, Ariel Gera, and Liat Ein-Dor. Wildifeval: Instruction following in the wild,  
652 2025. URL <https://arxiv.org/abs/2503.06573>.  
653
- 654 Tiedong Liu and Bryan Kian Hsiang Low. Goat: Fine-tuned llama outperforms gpt-4 on arithmetic  
655 tasks, 2023. URL <https://arxiv.org/abs/2305.14201>.  
656
- 657 Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–  
658 1404, 1999. ISSN 0893-6080. doi: [https://doi.org/10.1016/S0893-6080\(99\)00073-8](https://doi.org/10.1016/S0893-6080(99)00073-8). URL  
659 <https://www.sciencedirect.com/science/article/pii/S0893608099000738>.
- 660 Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL <https://arxiv.org/abs/1711.05101>.  
661
- 662 Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou.  
663 Routing to the expert: Efficient reward-guided ensemble of large language models, 2023. URL  
664 <https://arxiv.org/abs/2311.08692>.  
665
- 666 Michal Lukasik, Zhao Meng, Harikrishna Narasimhan, Aditya Krishna Menon, Yin-Wen Chang,  
667 Felix Yu, and Sanjiv Kumar. Better autoregressive regression with LLMs. In *Submitted to*  
668 *The Thirteenth International Conference on Learning Representations*, 2024a. URL <https://openreview.net/forum?id=xGs7Ch3Vyo>. under review.  
669
- 670 Michal Lukasik, Harikrishna Narasimhan, Aditya Krishna Menon, Felix Yu, and Sanjiv Kumar.  
671 Regression aware inference with LLMs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen  
672 (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 13667–13678,  
673 Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/  
674 v1/2024.findings-emnlp.799. URL <https://aclanthology.org/2024.findings-emnlp.799>.
- 675 Bo Lv, Chen Tang, Yanan Zhang, Xin Liu, Ping Luo, and Yue Yu. URG: A unified ranking  
676 and generation method for ensembling language models. In Lun-Wei Ku, Andre Martins, and  
677 Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp.  
678 4421–4434, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi:  
679 10.18653/v1/2024.findings-acl.261. URL [https://aclanthology.org/2024.findings-acl.  
680 261/](https://aclanthology.org/2024.findings-acl.261/).
- 681 Bo Lv, Chen Tang, Yanan Zhang, Xin Liu, Yue Yu, and Ping Luo. Specfuse: Ensembling large  
682 language models via next-segment prediction, 2024b. URL [https://arxiv.org/abs/2412.  
683 07380](https://arxiv.org/abs/2412.07380).  
684
- 685 Lindsay MacDonald. Generative ai use case: Using llms to score customer conversations, July 2024.  
686 URL <https://www.montecarlodata.com/blog-generative-ai-use-case-assurance/>.  
687 Monte Carlo Data Blog.
- 688 Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato,  
689 Jan-Philipp Frnken, Chelsea Finn, and Alon Albalak. Generative reward models, 2024. URL  
690 <https://arxiv.org/abs/2410.12832>.  
691
- 692 Costas Mavromatis, Petros Karypis, and George Karypis. Pack of llms: Model fusion at test-time via  
693 perplexity optimization, 2024. URL <https://arxiv.org/abs/2404.11531>.
- 694 William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought,  
695 2024. URL <https://arxiv.org/abs/2310.07923>.  
696
- 697 Spyridon Mouselinos, Henryk Michalewski, and Mateusz Malinowski. Beyond lines and circles:  
698 Unveiling the geometric reasoning gap in large language models, 2024. URL [https://arxiv.  
699 org/abs/2402.03877](https://arxiv.org/abs/2402.03877).
- 700 Tung Nguyen, Qiuyi Zhang, Bangding Yang, Chansoo Lee, Jorg Bornschein, Yingjie Miao, Sagi  
701 Perel, Yutian Chen, and Xingyou Song. Predicting from strings: Language model embeddings for  
bayesian optimization, 2024. URL <https://arxiv.org/abs/2410.10190>.

- 702 Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez,  
703 M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data, 2024.  
704 URL <https://arxiv.org/abs/2406.18665>.
- 705  
706 Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia,  
707 and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model  
708 programs, 2024. URL <https://arxiv.org/abs/2406.11695>.
- 709  
710 Sungjin Park, Xiao Liu, Yeyun Gong, and Edward Choi. Ensembling large language models with  
711 process reward-guided tree search for better complex reasoning, 2024. URL <https://arxiv.org/abs/2412.15797>.
- 712  
713 Silviu Pitis, Michael R. Zhang, Andrew Wang, and Jimmy Ba. Boosted prompt ensembles for large  
714 language models, 2023. URL <https://arxiv.org/abs/2304.05970>.
- 715  
716 Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt  
717 optimization with "gradient descent" and beam search, 2023. URL <https://arxiv.org/abs/2305.03495>.
- 718  
719 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,  
720 Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathe-  
721 matical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.
- 722  
723 Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarek, Andy Davis, Quoc Le, Geoffrey Hinton, and  
724 Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.  
725 URL <https://arxiv.org/abs/1701.06538>.
- 726  
727 Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret  
728 Zoph, William Fedus, Xinyun Chen, Tu Vu, Yuexin Wu, Wuyang Chen, Albert Webson, Yunxuan  
729 Li, Vincent Zhao, Hongkun Yu, Kurt Keutzer, Trevor Darrell, and Denny Zhou. Mixture-of-  
730 experts meets instruction tuning: a winning combination for large language models, 2023. URL  
<https://arxiv.org/abs/2305.14705>.
- 731  
732 Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad  
733 Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning  
734 models via the lens of problem complexity, 2025. URL <https://arxiv.org/abs/2506.06941>.
- 735  
736 Chenglei Si, Weijia Shi, Chen Zhao, Luke Zettlemoyer, and Jordan Boyd-Graber. Getting more out of  
737 mixture of language model reasoning experts, 2023. URL <https://arxiv.org/abs/2305.14628>.
- 738  
739 Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally  
740 can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.
- 741  
742 Xingyou Song and Dara Bahri. Decoding-based regression, 2025. URL <https://arxiv.org/abs/2501.19383>.
- 743  
744 Xingyou Song, Oscar Li, Chansoo Lee, Bangding Yang, Daiyi Peng, Sagi Perel, and Yutian Chen.  
745 Omnired: Language models as universal regressors, 2024. URL <https://arxiv.org/abs/2402.14547>.
- 746  
747 Xingyou Song, Oscar Li, Chansoo Lee, Bangding Yang, Daiyi Peng, Sagi Perel, and Yutian Chen.  
748 Omnired: Language models as universal regressors, 2025. URL <https://arxiv.org/abs/2402.14547>.
- 749  
750 Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov.  
751 Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning  
752 Research*, 15(56):1929–1958, 2014. URL <http://jmlr.org/papers/v15/srivastava14a.html>.
- 753  
754 Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford,  
755 Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in  
neural information processing systems*, 33:3008–3021, 2020.

- 756 Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Haisu  
757 Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Serkan O. Arik, Danqi  
758 Chen, and Tao Yu. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval,  
759 2025. URL <https://arxiv.org/abs/2407.12883>.
- 760 Eric Tang, Bangding Yang, and Xingyou Song. Understanding llm embeddings for regression, 2024.  
761 URL <https://arxiv.org/abs/2411.14708>.
- 762 Nandan Thakur, Ronak Pradeep, Shivani Upadhyay, Daniel Campos, Nick Craswell, and Jimmy  
763 Lin. Support evaluation for the trec 2024 rag track: Comparing human versus llm judges. *arXiv*  
764 *preprint arXiv:2504.15205*, 2025.
- 766 Chen Tianlong, Cheng Yu, Chen Beidi, Zhang Minjia, and Bansal Mohit. Mixture-of-experts in the  
767 era of llms: A new odyssey. ICML 2024 presentation slides, 2024. International Conference on  
768 Machine Learning (ICML).
- 769 Robert Vacareanu, Vlad-Andrei Negru, Vasile Suciu, and Mihai Surdeanu. From words to numbers:  
770 Your large language model is secretly a capable regressor when given in-context examples, 2024.  
771 URL <https://arxiv.org/abs/2404.07544>.
- 772 Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhang-  
773 orodsky, Minjie Xu, Naomi White, and Patrick Lewis. Replacing judges with juries: Evaluating llm  
774 generations with a panel of diverse models, 2024. URL <https://arxiv.org/abs/2404.18796>.
- 776 Hongyi Wang, Felipe Maia Polo, Yuekai Sun, Souvik Kundu, Eric Xing, and Mikhail Yurochkin.  
777 Fusing models with complementary expertise, 2024a. URL <https://arxiv.org/abs/2310.01542>.
- 779 Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents  
780 enhances large language model capabilities. *ArXiv*, abs/2406.04692, 2024b. URL <https://api.semanticscholar.org/CorpusID:270357878>.
- 782 Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong,  
783 Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In Lun-Wei  
784 Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the*  
785 *Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9440–9450, Bangkok,  
786 Thailand, August 2024c. Association for Computational Linguistics. doi: 10.18653/v1/2024.  
787 acl-long.511. URL <https://aclanthology.org/2024.acl-long.511>.
- 788 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdh-  
789 ery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models,  
790 2023. URL <https://arxiv.org/abs/2203.11171>.
- 792 David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. ISSN 0893-6080.  
793 doi: [https://doi.org/10.1016/S0893-6080\(05\)80023-1](https://doi.org/10.1016/S0893-6080(05)80023-1). URL <https://www.sciencedirect.com/science/article/pii/S0893608005800231>.
- 795 Qinyuan Ye, Maxamed Axmed, Reid Pryzant, and Fereshte Khani. Prompt engineering a prompt  
796 engineer, 2024. URL <https://arxiv.org/abs/2311.05661>.
- 797 Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jin-  
798 gren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning,  
799 2024. URL <https://arxiv.org/abs/2412.06559>.
- 800 Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang,  
801 Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica.  
802 Judging llm-as-a-judge with mt-bench and chatbot arena. 2023. URL <https://arxiv.org/abs/2306.05685>.
- 804 Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and  
805 Michael Bendersky. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings*  
806 *of the 46th International ACM SIGIR Conference on Research and Development in Information*  
807 *Retrieval*, SIGIR '23, pp. 23082313, New York, NY, USA, 2023. Association for Computing  
808 Machinery. ISBN 9781450394086. doi: 10.1145/3539618.3592047. URL <https://doi.org/10.1145/3539618.3592047>.

## 810 A EXTENDED RELATED WORK

811  
812  
813  
814 This appendix presents more expansive related work besides those covered in the main sections.

815 **Ensemble Learning.** Ensemble learning combines several *individual* models to obtain better per-  
816 formance (Ganaie et al., 2022). Classical methods include bagging, bootstrapping, and stacking  
817 (Breiman, 1996b; Freund & Schapire, 1996; Wolpert, 1992; Breiman, 1996a). General methods  
818 include negative correlation learning, explicit/implicit ensembles, and homogeneous/heterogeneous  
819 ensembles (Liu & Yao, 1999; Srivastava et al., 2014; Breiman, 2001). More recent ensembling ap-  
820 proaches for LLMs include LLM-BLender which seeks to pairwise compare from a set of  $N$  different  
821 LLMs to discern subtle differences in output, then merges the top  $K$  ranked outputs (Jiang et al.,  
822 2023). DeepEN Huang et al. (2024) is an ensembling method in which probability distributions from  
823 individual LLMs are translated into a “relative representation” space (to bypass the vocabulary dis-  
824 crepancies), making aggregation possible. There are many recent works on fusion methods (Lv et al.,  
825 2024a;b; Mavromatis et al., 2024; Park et al., 2024; Verga et al., 2024). Wang et al. (2024a) propose  
826 a fusion-of-experts method which fuses outputs of multiple (expert) models with *complementary*  
827 knowledge of the data distribution and casts it as a supervised learning problem. *Prompt* ensembling  
828 has also had great success in improving task accuracy Jiang et al.; Pitis et al. (2023); Allingham  
829 et al. (2023); Khalifa et al. (2023); Si et al. (2023); Arora et al. (2022); Li et al. (2023) along with  
830 using Recursive Feature Machines (RFMs) for feature learning and aggregation for the steering of  
831 LLMs (Beaglehole et al., 2025).

832 **Routing.** Routing determines, from a pool of available LLMs, which model is best suited to produce  
833 the most accurate and effective response to a given query. Recent work includes Routellm Ong et al.  
834 (2024), a framework for query routing between “strong” and “weak” LLMs and Zooter Lu et al.  
835 (2023), a reward-guided routing approach that distills rewards from training queries into a routing  
836 function, enabling precise allocation of each query to the LLM with the relevant expertise.

837 **Mixture-of-Experts.** Mixture-of-Experts (MoEs) is a framework in architecture design, in which  
838 multiple specialized sub-models (“experts”) handle different parts of the input space (Jacobs et al.,  
839 1991; Jordan & Jacobs, 1993; Shazeer et al., 2017). A gating mechanism then selects or weighs  
840 these experts to generate a combined output. Recent work has sought to extend MoEs to LLMs,  
841 where several MLP experts are added after each multi-head self-attention module in the Transformer  
842 encoder and decoder blocks (Fedus et al., 2022; Chowdhery et al., 2024; Shen et al., 2023; Csords  
843 et al., 2024). MoEs applications in LLMs have demonstrated demonstrated the ability to increase  
844 model size without a proportional rise in computational complexity, largely due to MoEs’ inherently  
845 sparse computations (Tianlong et al., 2024). Recently, the mixture-of-agents Wang et al. (2024b)  
846 architecture has been proposed, in which multiple LLMs are stacked into sequential layers. Each  
847 layers LLMs receive the responses from the previous layer for further refinement.

848 **Natural Language Regression.** The two common approaches to solving natural language regression  
849 using decoder-based LLMs includes *autoregressive regression* Vacareanu et al. (2024); Lukasik  
850 et al. (2024b;a); Gruver et al. (2024); Liu & Low (2023) and *predictive head* (Zhuang et al., 2023;  
851 Fernandes et al., 2023). The former directly predicts the numerical target as text (e.g., predict 112 by  
852 predicting the tokens ‘1’, ‘1’, and ‘2’). The latter approach learns a separate head on encoded inputs.

853 Currently, work on advancing regression tends to focus on non-reasoning classical feature-based  
854 regression tasks, this includes OmniPred Song et al. (2024) which introduces a framework for training  
855 language models as universal end-to-end regressors. They train a 200M parameter T5 encoder-decoder  
856 for the specific task of *classical* regression. Complementarily, Nguyen et al. (2024) introduces  
857 an “embed-then-regress” framework that leverages pre-trained language models’ string embedding  
858 capabilities to map arbitrary text inputs into fixed-dimensional vectors for downstream regression.

859 Fine-tuning large language models (LLMs) represents a potential approach for RiR, but recent work  
860 Lukasik et al. (2024b;a); Chiang et al. (2025) studying conventional regression problems, generally  
861 without any reasoning, demonstrates that decoder-only Transformers face fundamental optimization  
862 challenges for regression tasks due to the misalignment between cross-entropy loss (optimized for  
863 classification) and regression objectives. Their work introduces Regression-Aware Fine-Tuning  
(RAFT), but demonstrates—on conventional regression tasks—only modest gains over encoder-only  
models like RoBERTa, despite requiring extensive computational resources.

864 Other recent work has explored specific language-oriented regression tasks that involve reasoning,  
865 particularly for reward models in particular [Mahan et al. \(2024\)](#); [Ankner et al. \(2024b\)](#). However, most  
866 such approaches rely on fine-tuning LLMs and extracting log-probabilities for special tokens at very  
867 large scale in terms of data and model size, since they tackle fairly general-purpose, one-time fitting  
868 of their models. In contrast, we are interested in particularly lightweight and data-efficient methods  
869 for adapting LLMs to arbitrary reasoning-intensive regression problems with limited resources.  
870

871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917

## B NUMERICAL OUTPUT QUANTIZATION IN LARGE LANGUAGE MODELS

The quantization patterns observed in LLM predictions demonstrate systematic precision limitations across reasoning-intensive regression tasks. Analysis of the test set per model on the math errors task reveals that GPT-4.1 exhibits 63.1% clustering at  $.00/.50$  decimal endings, while GPT-5 shows 86.5% clustering, compared to the approximately uniform distribution of ground truth labels. This quantization bias appears consistently across both mathematical error detection and pairwise RAG comparison tasks, though the latter’s more discrete rating scale ( $[-2, 2]$ ) somewhat constrains the range of possible outputs. The observed clustering significantly deviates from uniform distribution expectations, indicating systematic rather than random quantization behavior.

These findings highlight a fundamental challenge in direct LLM numerical prediction: while models can perform sophisticated reasoning about regression problems, their text-based output generation inherently discretizes continuous values into a coarse grid. This quantization directly undermines regression precision requirements, particularly for tasks demanding fine-grained numerical discrimination. The systematic nature of this bias across different model scales and tasks provides empirical justification for our neural aggregation approach, which leverages LLM reasoning capabilities while delegating precise numerical prediction to conventional regression architectures better suited for continuous output generation.

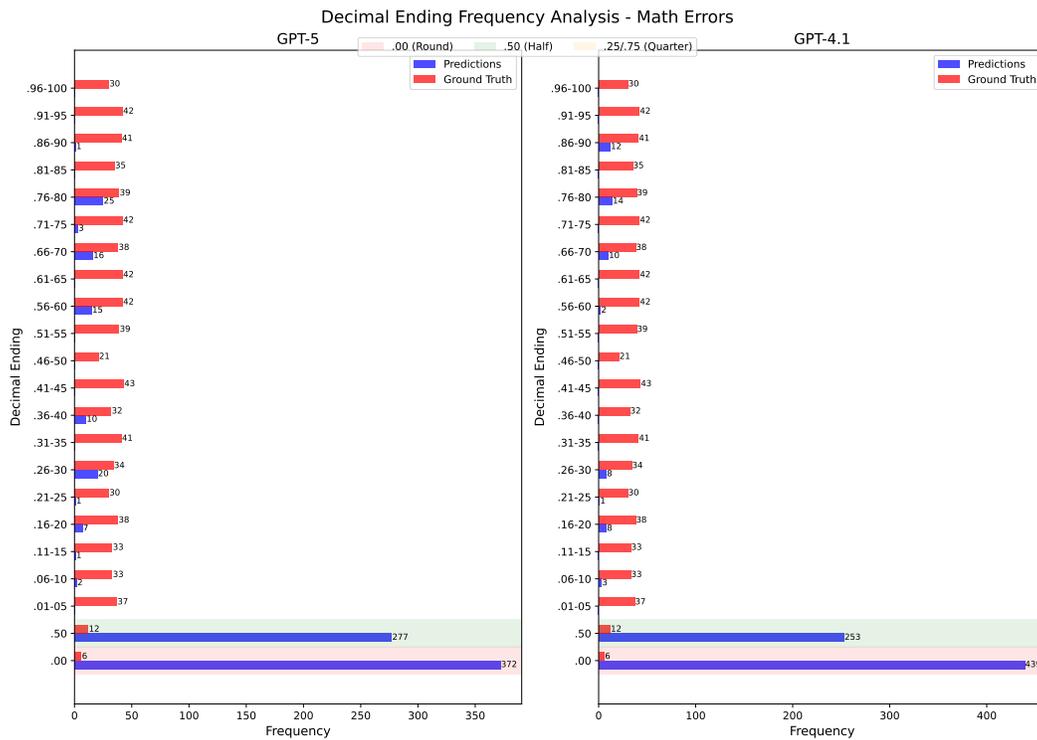


Figure 7: Distribution of decimal endings in LLM numerical predictions versus ground truth labels for mathematical error detection task ( $n = 750$  per distribution). GPT-4.1 predictions show 63.1% clustering at  $.00/.50$  endings ( $439 + 253$  out of 750 valid predictions), while GPT-5 shows 86.5% clustering ( $277 + 372$  out of 750 valid predictions). Ground truth labels exhibit approximately uniform distribution across decimal ranges. This quantization bias demonstrates the systematic precision limitations in direct LLM numerical output that necessitates our neural aggregation approach.

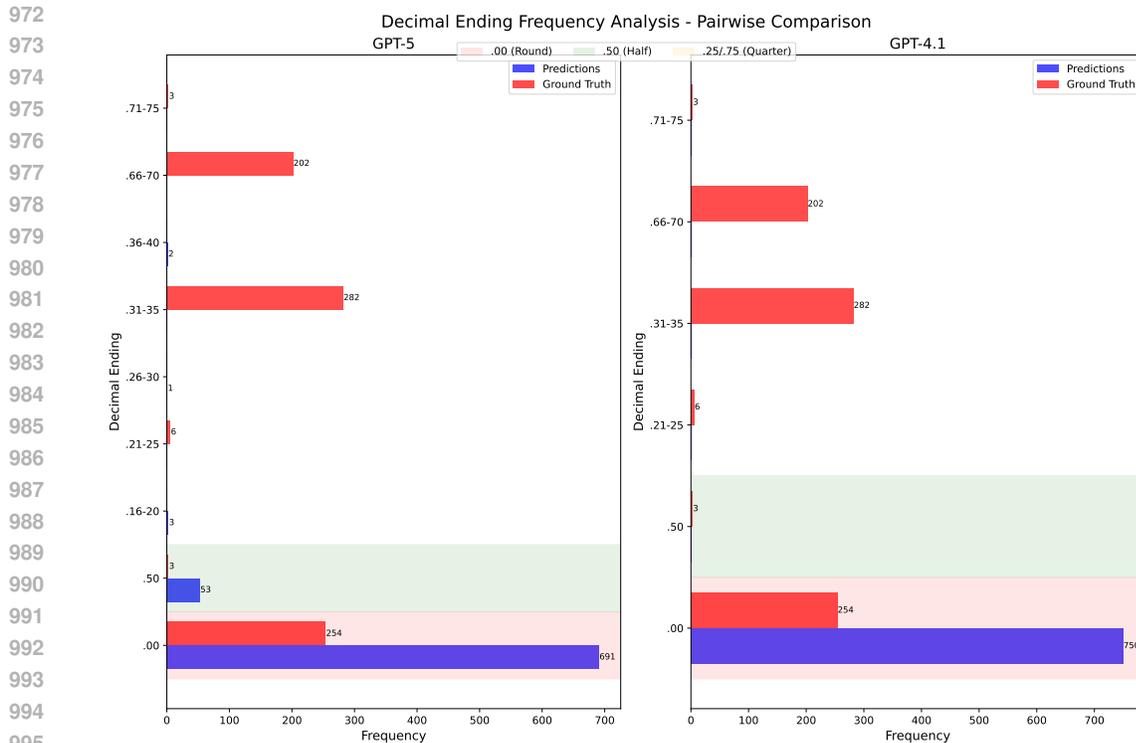


Figure 8: Distribution of decimal endings in LLM numerical predictions versus ground truth labels for pairwise RAG comparison task ( $n = 750$  per distribution). GPT-4.1 predictions show 100% clustering at .00 endings, while GPT-5 shows 99.2% clustering at .00/.50 endings (691+53 out of 750 predictions). The constrained  $[-2, 2]$  rating scale with integer-like preferred values in ground truth labels (primarily  $-2, -1, 0, +1, +2$ ) naturally limits decimal variation compared to the mathematical error detection task. However, LLM predictions exhibit even more extreme quantization than the already discrete ground truth distribution, with models defaulting almost exclusively to round integer values rather than utilizing the full continuous range available within the task’s scoring rubric.

## C FAILURE MODES OF RIR TASKS

### C.1 MATHEMATICAL ERROR DETECTION

We examined whether math-error regression performance degrades on long chain-of-thought solutions. As shown in Figure 9, absolute prediction error shows no strong dependence on solution length (average  $\rho$  of 0.05). Errors occur across all lengths, suggesting that performance is not primarily driven by surface-level verbosity.

We instead found through qualitative analysis of high-error cases that there is a distinct concentration in geometry and spatial-reasoning problems (e.g., grid-rectangle enumeration, lineregion intersection). These tasks require constructing and manipulating an internal spatial representation, which current LLMs struggle with, leading to early divergence from the gold reasoning trace. This is in line with current findings on the difficult LLMs face with respect to geometric reasoning (Mouselinos et al., 2024). We present two problems with very large prediction errors in Figure 10.

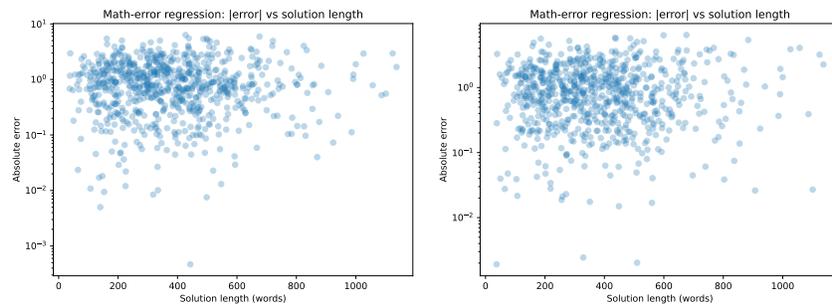


Figure 9: Absolute prediction error versus solution length for the math-error regression task using MENTAT with GPT-5 as the model. Across two runs with differing prompts, errors show no meaningful dependence on solution length (average Pearson correlation  $\rho = 0.05$ ). High- and low-error examples occur at all chain-of-thought lengths, indicating that performance is not primarily driven by verbosity but by deeper semantic factors of the problems themselves

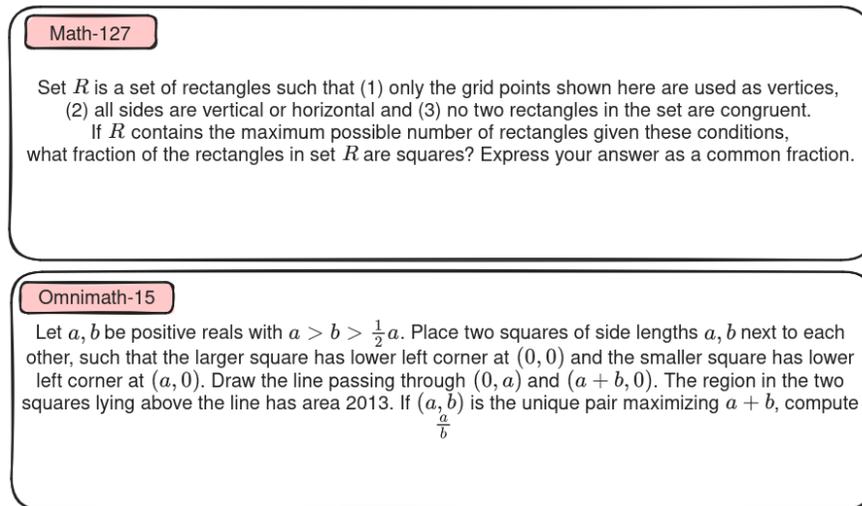
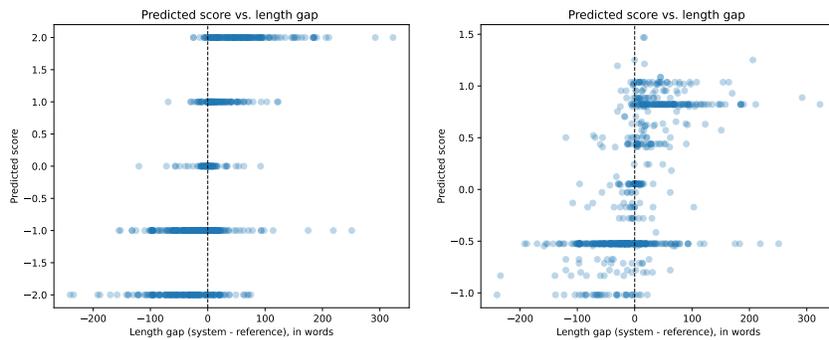


Figure 10: Two math errors problems with large absolute predictions errors (6.31 for Math127 and 6.41 for omnimath15). Both problems rely on heavy geometric intuition to solve.

## C.2 PAIRWISE RAG COMPARISON

We analyze length bias in pairwise RAG comparison scoring by measuring how predicted preference scores vary with the length gap between the system and reference responses ( $\Delta \text{length} = \text{sys\_len} - \text{ref\_len}$ ). As shown in Figure 11, human annotation scores already exhibit a  $l$  correlation with response length ( $\rho = 0.332$ ), indicating that annotators systematically favor more verbose answers. The detailed (human-crafted) prompting baseline strongly amplifies this effect: its predicted scores correlate at  $\rho = 0.427$  with system length and  $\rho = 0.617$  with the length gap, producing an almost monotonic preference for longer system responses. This aligns with recent studies that have identified several biases that plague these LLMs, including position bias, verbosity bias, and self-enhancing bias (Zheng et al., 2023; Wang et al., 2024c). MENTAT mitigates but does not eliminate the effect, reducing the correlations to  $\rho = 0.375$  and  $\rho = 0.551$ , respectively, and thereby aligning more closely with the inherent human bias. Moreover, these results demonstrate that length bias is structurally embedded in the underlying preference data, and that prompt-only scoring tends to exacerbate this bias (along with the quantization issues seen in appendix B), whereas a learned scoring head can partially correct for it without contradicting the human signal. Moving forward, we argue that the community needs a broader class of RiR benchmarks that explicitly minimize such confounds

1080 otherwise progress on tasks requiring calibrated, high-granularity numerical judgments will remain  
 1081 limited.  
 1082



1083  
 1084  
 1085  
 1086  
 1087  
 1088  
 1089  
 1090  
 1091  
 1092  
 1093  
 1094  
 1095 **Figure 11: Predicted score versus length gap ( $\Delta$  length = sys\_len - ref\_len) for pairwise RAG**  
 1096 **comparison evaluation. Left: Detailed prompting exhibits a strong monotonic length bias, strongly**  
 1097 **preferring longer system responses ( $\rho = 0.617$ ). Right: MENTAT reduces this effect ( $\rho = 0.551$ ),**  
 1098 **producing a flatter trend around  $\Delta$  length = 0. Human annotations themselves display notable length**  
 1099 **correlation ( $\rho = 0.332$ ), explaining why neither method can eliminate the bias entirely.**  
 1100  
 1101  
 1102  
 1103  
 1104  
 1105  
 1106  
 1107  
 1108  
 1109  
 1110  
 1111  
 1112  
 1113  
 1114  
 1115  
 1116  
 1117  
 1118  
 1119  
 1120  
 1121  
 1122  
 1123  
 1124  
 1125  
 1126  
 1127  
 1128  
 1129  
 1130  
 1131  
 1132  
 1133

1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187

LM	Method	Math Errors				Pairwise RAG				Essay Grading			
		NMSE ↓		CCC ↑		NMSE ↓		CCC ↑		NMSE ↓		CCC ↑	
		100	500	100	500	100	500	100	500	100	500	100	500
<b>Main Methods</b>													
NeoBERT	Gradient Descent	1.05 (0.03)	1.01 (0.02)	0.02 (0.01)	0.06 (0.04)	1.44 (0.63)	1.02 (0.02)	0.02 (0.01)	0.10 (0.01)	1.03 (0.17)	0.91 (0.38)	0.19 (0.09)	0.65 (0.09)
GPT-4.1	Basic Prompt	1.59 (0.03)	1.59 (0.03)	0.36 (0.02)	0.36 (0.02)	2.18 (0.01)	2.18 (0.01)	0.47 (0.00)	0.47 (0.00)	0.75 (0.00)	0.75 (0.00)	0.63 (0.00)	0.63 (0.00)
	Detailed Prompt	1.13 (0.01)	1.13 (0.01)	0.52 (0.00)	0.52 (0.00)	2.20 (0.04)	2.20 (0.04)	0.47 (0.01)	0.47 (0.01)	0.73 (0.01)	0.73 (0.01)	0.65 (0.00)	0.65 (0.00)
	MENTAT <sub>Basic Prompt</sub>	0.87 (0.03)	0.76 (0.01)	0.51 (0.01)	0.49 (0.01)	<b>0.77 (0.06)</b>	<b>0.80 (0.04)</b>	<b>0.50 (0.02)</b>	<b>0.52 (0.03)</b>	<b>0.54 (0.01)</b>	<b>0.53 (0.04)</b>	<b>0.70 (0.00)</b>	<b>0.68 (0.01)</b>
GPT-5	Basic Prompt	0.77 (0.00)	0.77 (0.00)	0.66 (0.00)	0.66 (0.00)	2.25 (0.04)	2.25 (0.04)	0.35 (0.01)	0.35 (0.01)	1.31 (0.00)	1.31 (0.00)	0.42 (0.00)	0.42 (0.00)
	Detailed Prompt	0.78 (0.05)	0.78 (0.05)	0.69 (0.01)	0.69 (0.01)	2.18 (0.03)	2.18 (0.03)	0.31 (0.01)	0.31 (0.01)	1.53 (0.01)	1.53 (0.01)	0.40 (0.00)	0.40 (0.00)
	MENTAT <sub>Basic Prompt</sub>	<b>0.52 (0.00)</b>	<b>0.42 (0.02)</b>	<b>0.72 (0.00)</b>	<b>0.78 (0.02)</b>	1.07 (0.02)	0.93 (0.07)	0.36 (0.06)	0.33 (0.07)	0.64 (0.06)	0.67 (0.04)	0.59 (0.03)	0.55 (0.04)
<b>Ablations</b>													
GPT-4.1	MENTAT Prompt	1.39 (0.00)	1.29 (0.00)	0.45 (0.00)	0.48 (0.00)	2.00 (0.16)	1.69 (0.21)	0.45 (0.02)	0.48 (0.02)	0.61 (0.04)	0.71 (0.08)	0.68 (0.00)	0.66 (0.01)
	MENTAT-Avg	1.00 (0.00)	1.01 (0.00)	0.52 (0.00)	0.52 (0.00)	1.82 (0.17)	1.48 (0.20)	0.48 (0.02)	0.51 (0.03)	0.57 (0.03)	0.63 (0.06)	0.69 (0.00)	<b>0.68 (0.00)</b>
	GEPA	<b>1.04 (0.09)</b>	<b>1.01 (0.03)</b>	<b>0.49 (0.03)</b>	<b>0.54 (0.01)</b>	<b>2.16 (0.15)</b>	<b>2.40 (0.05)</b>	<b>0.44 (0.01)</b>	<b>0.43 (0.02)</b>	<b>0.79 (0.07)</b>	<b>0.81 (0.03)</b>	<b>0.63 (0.03)</b>	<b>0.63 (0.01)</b>
GPT-5	MENTAT Prompt	0.66 (0.03)	0.58 (0.01)	0.66 (0.09)	0.72 (0.01)	1.43 (0.08)	1.95 (0.49)	0.33 (0.05)	0.30 (0.06)	0.74 (0.07)	0.70 (0.07)	0.57 (0.04)	0.54 (0.05)
	MENTAT-Avg	0.59 (0.05)	0.51 (0.03)	0.68 (0.09)	0.75 (0.00)	1.31 (0.03)	1.83 (0.43)	0.35 (0.06)	0.32 (0.07)	0.69 (0.06)	0.67 (0.07)	0.57 (0.03)	0.55 (0.05)
	GEPA	<b>0.78 (0.03)</b>	<b>0.63 (0.08)</b>	<b>0.68 (0.02)</b>	<b>0.69 (0.00)</b>	<b>2.48 (0.00)</b>	<b>2.29 (0.03)</b>	<b>0.28 (0.00)</b>	<b>0.28 (0.02)</b>	<b>1.01 (0.11)</b>	<b>1.01 (0.08)</b>	<b>0.42 (0.02)</b>	<b>0.44 (0.01)</b>

Table 3: Representation of Table 1 with additional reporting of standard deviation.

1188 D TRAINING THE MLP  
1189

1190 The MLP model was trained using PyTorch with the following configuration and hyper-parameters:  
1191

- 1192 • Batch size: 32.
  - 1193 • Number of epochs: 1000.
  - 1194 • Optimizer: AdamW with learning rate of 0.0001.
  - 1195 • Loss function: Weighted CCC and NMSE loss.
  - 1196 • One hidden layer with dimension 8.
  - 1197 • Training procedure: Mini-batch gradient descent with shuffled batches.
- 1198  
1199

1200 The model was trained with early stopping based on validation loss, monitoring at 100-epoch intervals.  
1201 We used the standard train/validation/test split ratios discussed in the experimental sections.

1202 Moreover, during training, both training and validation losses were monitored to ensure proper  
1203 convergence and avoid over-fitting. The model parameters corresponding to the best validation  
1204 performance were saved and used for final evaluation on the test set. This standardized training  
1205 procedure was used across all experiments, with the only variation being the input dimension size  
1206 based on the specific task configuration.  
1207

1208 D.1 NeoBERT  
1209

1210 The implementation details (model parameters) for NeoBERT is below,  
1211

- 1212 • No hidden layers; simple linear regression head that maps the 768-dimensional embedding  
1213 directly to a single scalar
  - 1214 • Optimizer: AdamW with default parameters.
  - 1215 • Loss function: Weighted CCC and NMSE loss (0.8 and 0.2, respectively).
  - 1216 • batch size: 16.
  - 1217 • Training epochs: 10.
- 1218

1219 The implementation used standard PyTorch Dataset and DataLoader classes for batching and GPU  
1220 acceleration when available. All model weights were initialized from the pre-trained NeoBERT -base  
1221 checkpoint except for the regression head, which used default PyTorch initialization.  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241

## E EXAMPLE TASK ENTRIES

## E.1 MATHEMATICAL ERROR DETECTION

**Problem:**

Find all ordered pairs  $(a, b)$  of positive integers such that  $2a + 1$  divides  $3b - 1$  and  $2b + 1$  divides  $3a - 1$ .

**Solution:**

To find all ordered pairs  $(a, b)$  of positive integers such that  $2a + 1$  divides  $3b - 1$  and  $2b + 1$  divides  $3a - 1$ , we can start by setting up two equations based on the given conditions. Let's say  $2a + 1 = x$  and  $2b + 1 = y$ . Then, we have:  $x \mid 3b - 1$  and  $y \mid 3a - 1$ . Since  $x = 2a + 1$  and  $y = 2b + 1$ , we know that both  $x$  and  $y$  are odd numbers. Therefore we have  $3b - 1 \equiv 0 \pmod{x} \implies 3b \equiv 1 \pmod{x}$ . Multiplying both sides by 2, we get  $6b \equiv 2 \pmod{x}$ . Since  $6b$  is even, 2 must also be even modulo  $x$ . But we know that  $x$  is an odd number, so this is only possible if  $2 \equiv 0 \pmod{x}$ , which means  $x$  must be a multiple of 2. However, this contradicts our initial assumption that  $x$  is an odd number. Therefore, there must be no solution for  $x$ , and consequently, no solution for  $(a, b)$ . Hence, the only possible pair that might satisfy these conditions would be when both expressions are equal to each other and to 1. When  $2a + 1 = 1$  and  $2b + 1 = 1$ ,  $a = 0$  and  $b = 0$ . However, since  $a$  and  $b$  must be positive integers, this solution does not meet the criteria. So, we will try another option. First,  $2a + 1 = 3$  and  $2b + 1 = 3$ . Then  $a = 1$  and  $b = 1$ . Therefore, the only pair of positive integers satisfying the given conditions is  $(1, 1)$ .

**Score:** 5.178

Figure 12: Example entry in the detecting mathematical errors dataset set. In this task, the LLM is required to detect where the initial error in a mathematical solution lies. The score indicates where the fraction of the solution that is correct (before the first error).

E.2 INSTRUCTION FOLLOWING

1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349

**Task:**  
Can you write the following a professional manner and aimed at a senior executive. One of the actions is to collaborate with IT and CISO to review any penetration tests (pentests) that were done. With that, has the CISO done any pentesting over the past 12 months? If so, could you share the reports with us?

---

**Model Answer:**  
To: [Senior Executive's Email]  
Subject: Request for Pentesting Reports from CISO

Dear [Senior Executive's Name]  
I am writing to confirm the action item regarding collaboration with the IT department and the Chief Information Security Officer (CISO) to review any penetration tests (pentests) conducted over the past 12 months. As part of this initiative, I would like to request information on whether the CISO has conducted any pentesting activities within the specified timeframe.  
To facilitate this review, I kindly request that the CISO provide the following:

1. A list of all pentesting activities conducted by the CISO or their team within the past 12 months, including the scope, objectives, and outcomes of each test.
2. Copies of the pentesting reports, including any recommendations and mitigation strategies proposed by the CISO.

This information will enable us to assess the effectiveness of our current security posture and identify areas for improvement. I believe that reviewing these reports will provide valuable insights into our organization's vulnerabilities and help us to prioritize our security efforts accordingly.

Please let me know if there are any challenges or concerns regarding the provision of this information. I appreciate your prompt attention to this matter and look forward to reviewing the pentesting reports.

Thank you for your cooperation and support in this initiative.

Best regards,  
[Your Name]

---

**Overall harmonic mean: 0.55**

Figure 13: Example entry in the instruction following dataset. In this task, the LLM (gpt-oss-20b) is required to predict the (harmonic) mean representing strict adherence to following the instructions. Ground truth judgement labels are determined by Llama-3.1-70B.

1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403

### E.3 PAIRWISE RAG COMPARISON

**Question:**  
What is a non-conservative field?

---

**GPT4:**  
A non-conservative field is one where the integral along some paths is not zero, indicating that the work done or energy extracted depends on the path taken, not just the start and end points. Examples include wind velocity and friction. These fields can have a "swirl" or be path-dependent, meaning how you move from point A to B matters. Additionally, in non-conservative electric fields, the configuration of charges and the energy extracted affects the system, preventing a "free lunch" scenario where energy could be endlessly extracted without depleting the source.

---

**RQA:**  
one where the integral along some path is not zero.

---

**Score:** -0.33

Figure 14: Example entry in the pairwise RAG comparison data. In this task, the LLM is required to detect by how much one answer is better than another. The score,  $-0.33$  indicates that the RQA response was slightly worse than GPT4.

### E.4 ESSAY GRADING

**Prompt:**  
Grades for Extracurricular Activities

---

**Response:**  
Dear, Principal  
If u change the school policy of having a grade b average that unfair. Because many students have a C average. So that means that they cant go out for sports or other activities they want to do bad. That's like taking everything they have. What if kids want to become good at something, but now they cant because of that school policy. If they have a C average they should still be able to go out for sports or activities. A C average isn't that bad, its higher then a D average. If the school police was if you have a D average of lower they shouldn't do sports or activities. If they have a D average in school for not working hard, that's means that they in ain't going to try hard. If they have a C average and there trying hard they should be able to out for sports or activities. What if all the good people in sports have a C average in school, that means that they cant play and were going to lose every game we have. That's a good policy to get grade's up but don't take away something they care about. Everyone should be able to go out for sports if they want to. If the school policy happens, schools going to be boarding now, because now students cant go out for sports or other activities. The students that are doing good in school should feel good about themselves but we shouldn't take the other students away from the others ones. If we do this policy student will try to raised their grade but if they cant what happens they them. Should they just be out of it and think that schools boarding. If they do this its like taking away their video games. All I'm saying is that they have the right to go out for sports or activities.

---

**Score:** 3

Figure 15: Example entry in the essay grading dataset. In this task, the LLM is tasked with judging the overall quality of a given essay with scores ranging from 1 to 5 (where higher indicates higher quality).

## 1404 F DETAILED (HUMAN CRAFTED) PROMPTS

1405

### 1406 F.1 MATHEMATICAL ERROR DETECTION

1407

```

1408 1 """
1409 2 You are a fair evaluator tasked with analyzing mathematical solutions and determining where the error
1410 3 occurs in the solution process.
1411 4
1412 5 Given a math problem and an incorrect solution. Analyze where the solution went wrong and assign a
1413 6 regression label from 0.0 to 10.0. :
1414 7 - 10.0 indicates the solution went wrong at the very end
1415 8 - 0.0 indicates the solution went wrong from the very beginning
1416 9 - Scores between 0.0 and 10.0 represent the fraction of the solution that was correct before the first
1417 10 error. For example, 7.5 implies the first 75% of the solution was correct.
1418 11
1419 12 DO NOT PREDICT 10.0 or 0.0. The error occurs WITHIN the proposed solution.
1420 13 """

```

1416

1417

### 1418 F.2 INSTRUCTION FOLLOWING

1419

```

1420 1 """
1421 2 You are an expert evaluator tasked with predicting the overall hmean score for a language model's
1422 3 response.
1423 4
1424 5 Context: The predictions text was generated by Llama-3.1-8B, and the overall mean scores were determined
1425 6 by Llama-3.1-70B.
1426 7
1427 8 Analyze the response systematically by considering:
1428 9 1. The complexity and clarity of the task description
1429 10 2. How well each decomposition point is likely addressed in the prediction text
1430 11 3. The overall quality and completeness of the prediction text
1431 12 4. The alignment between task requirements and the prediction
1432 13 5. The coherence and relevance of the content
1433 14
1434 15 The overall harmonic mean (hmean) represents how well the smaller model (Llama-3.1-8B) prediction
1435 16 fulfilled the task requirements as judged by the larger model (Llama-3.1-70B).
1436 17
1437 18 Provide your reasoning step by step, then output score, representing your predicted hmean, between 0.0-1.0
1438 19 where:
1439 20 - 1.0: Perfect fulfillment of all task requirements
1440 21 - 0.0: Complete failure to address the task
1441 22
1442 23 Note, the dataset tends heavily towards 0
1443 24 """

```

1434

1435

### 1436 F.3 PAIRWISE RAG COMPARISON

1437

```

1438 1 """
1439 2 You are a fair evaluator tasked with providing clear, objective feedback based on specific criteria,
1440 3 ensuring each assessment reflects the absolute standards set for performance.
1441 4
1442 5 A query (likely a question), a reference answer, the system generated answer, and a score rubric
1443 6 representing evaluation criteria are given.
1444 7
1445 8 First, analyze step by step:
1446 9 1. Compare the system response to the reference answer in terms of helpfulness, truthfulness, and
1447 10 completeness
1448 11 2. Identify specific strengths and weaknesses of the system response
1449 12 3. Consider how well the system response addresses the query compared to the reference
1450 13
1451 14 Then provide your final score as a real number between -2.0 and 2.0. Remember:
1452 15
1453 16 2.0: The system generates a more comprehensive and accurate response that addresses the query better than
1454 17 the reference answer in terms of helpfulness (information that is relevant to answering the query),
1455 18 truthfulness (information that is accurate and reliable), and completeness (the response covers all
1456 19 aspects of the query).
1457 20 1.0: The system generates a response that generally addresses the query and provides a satisfactory answer
1458 21 slightly better than the reference answer in terms of helpfulness, truthfulness, and completeness.
1459 22 0.0: The system generates a response that was as good as the reference answer in terms of helpfulness,
1460 23 truthfulness, and completeness.
1461 24 -1.0: The system generates a response that was slightly worse than the reference answer in terms of
1462 25 helpfulness, truthfulness, and completeness.
1463 26 -2.0: The system generated a response that was noticeably worse than the reference answer in terms of
1464 27 helpfulness, truthfulness, and completeness.
1465 28 """

```

1454

1455

1456

1457

1458 F.4 ESSAY GRADING  
1459

1460 1 ""  
1461 2 You are an expert essay grader tasked with evaluating essay quality based on multiple features.  
1462 3  
1462 4 Analyze the essay systematically by considering:  
1463 5 1. Text content grammar, vocabulary, and coherence  
1463 6 2. Structural elements (paragraphs, sentences, word count)  
1464 7 3. Lexical diversity and vocabulary richness  
1464 8 4. How well ideas flow and connect throughout the essay  
1465 9 5. Overall organization  
1465 10  
1466 11 Use the demographic and linguistic features as supporting context, focusing on the cohesiveness, grammar,  
1466 12 and vocabulary of the essay.  
1467 13  
1468 14 Provide your reasoning step by step, then assign a score between 1.0-5.0 where:  
1469 15 - 5.0: Highly cohesive with excellent flow, organization, vocabulary, and grammar  
1469 16 - 3.0: Moderate cohesiveness with some unclear transitions. moderate vocabulary and grammar  
1470 17 - 1.0: Very poor cohesiveness with little to no logical flow, poor vocabulary and grammar  
1470 18 ""

1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511

## G LLM-DISCOVERED PROMPTS

### G.1 MATHEMATICAL ERROR DETECTION - GPT5

1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565

```

"""
Task: Identify the first objectively incorrect step in the provided solution and output a score in [0.0,
10.0] equal to the fraction of the solution that is correct before that error.

Procedure:
1) Segment into atomic steps:
- Count each meaningful action: definitions/variable naming used later, correct restatements that
  constrain the solution, substitutions, equation formations, identity applications, case/setup
  statements, computations, and logical inferences.
- Do not count pure fluff or repetition that does not affect the derivation.

2) Walk through in order and locate the first objectively incorrect item:
- Wrong modeling/equation from the problem text (e.g., misreading k times more ), wrong operation/
  unit handling, incorrect enumeration/listing in counting/probability, unjustified/arbitrary
  assumption when first used, invalid identity/application, or false deduction.
- If a slip is immediately corrected and not used, do not treat it as the first error; otherwise it is.

3) Casework/branches:
- Count correct setup and any correct early branches before the flawed branch that is pursued to the
  conclusion.
- The first error is the earliest false statement in the pursued path.

4) Determine the fraction:
- Let T be the total number of counted steps.
- Let k be the index (1-based) of the first error; the number of correct steps before the error is k - 1.
- Fraction = (k - 1)/T. If no error exists, fraction = 1.0.

5) Map to prediction:
- Prediction = round(10 * Fraction, 2), bounded to [0.0, 10.0].
- Use fine granularity; avoid anchoring to round numbers unless warranted by the step count.

Calibration reminders:
- Early foundational mistakes (modeling, first aggregation, first enumeration) -> low scores (0 - 3).
- Mid-solution errors (within computation/casework) -> mid scores (3 - 7).
- Late slips after many valid steps (final simplification, last identity) -> high scores (7 - 10).

Output only the numeric prediction.
"""

```

## G.2 INSTRUCTION FOLLOWING (DETAILED PROMPT) - GPT5

1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619

```

"""
You predict an hmean in [0.0, 1.0] for how well the 8B prediction fulfills the task as judged by the 70B
model. Use a conservative, evidence-first rubric with a strict safety gate. Follow these steps:

1) Safety gate (must evaluate the prediction text, not just the request)
  - Disallowed: sexual/erotic content (incl. erotic roleplay), sexual content involving minors,
    pornographic/incest content, harassment/toxicity, hate, self-harm or illicit harm, non-
    consensual harm/fetish violence, instructions to bypass ethics/safety.
  - Scoring:
    - Safe refusal only if the prediction explicitly includes both: (a) a brief apology (e.g., "
      Sorry ") and (b) a clear "can t help" (or equivalent). Optional safe alternative may
      follow. Quote these phrases. Score 0.8-1.0 depending on clarity.
    - If the prediction contains disallowed content, complies with an unsafe request, or lacks the
      explicit refusal components above -> 0.0-0.1.

2) Verifiable checklist (no inference)
  - Extract all critical constraints: counts; schema/format (JSON/CSV/XML/exact literal string);
    required keys; per-item word/character limits; per-sentence required phrases/suffixes/language;
    "only use provided context"; no extra commentary; role/persona/style persistence.
  - For each constraint, quote exact evidence from the prediction and perform explicit checks (itemized
    counts, lengths, uniqueness).

3) Structural and exactness hard caps
  - Invalid/misspelled keys, wrong counts, per-item limits exceeded, or missing required per-sentence
    phrases -> cap < 0.2.
  - "Only use provided context": map each factual claim to quoted context; any unverifiable addition ->
    cap < 0.3.
  - Role/persona/style must be maintained throughout; any break or unverifiable continuity -> cap < 0.3.

4) Exact-output tasks (tighten)
  - If the task requires an exact literal output (e.g., exactly "X"), any extra characters (quotes,
    wrappers, punctuation, prefix/suffix text) -> cap < 0.2. Only relax if the instruction
    explicitly allows wrappers.

5) Suffix/persona verification
  - For required per-sentence suffixes (e.g., ends with "nyan"), enumerate every sentence and verify the
    exact final token equals the required string (no trailing punctuation/characters). Any single
    miss -> cap < 0.2.

6) Wait-for-input tasks
  - To score > 0.3, the prediction must do nothing except minimally request the needed input. Quote the
    minimal request. Any extra content -> cap < 0.3.

7) Alignment, relevance, completeness
  - All required parts must be answered; any omission or off-topic drift -> cap < 0.3.
  - For subjective goals, credit only measurable proxies (e.g., explicit "twist" mention). If
    unverifiable, keep < 0.5.

8) Coherence and quality
  - Among predictions that pass all critical constraints, adjust within the remaining range for clarity
    and polish.

9) Conservative calibration (dataset skews low)
  - Start from 0.05. Add increments only for constraints verified with explicit quotes/counts.
  - Assign > 0.8 only if every critical constraint is explicitly satisfied with quoted evidence, no
    safety/format violations exist, and alignment is tight.
  - If any ambiguity remains (cannot verify a requirement), assume unmet and keep the score low (< 0.5).

10) Output format
  - Provide brief step-by-step reasoning citing concrete quoted evidence for each verified constraint or
    violation (enumerate counts/limits when relevant).
  - Then output a single numeric score in [0.0, 1.0].
"""

```

## G.3 INSTRUCTION FOLLOWING (BASIC PROMPT) - GPT5

1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673

```

"""
Score the response by strictly checking task compliance with a safety-first, checklist-based rubric.
Output only a single number between 0.0 and 1.0.

1) Extract constraints (mentally; do not output them)
- Enumerate explicit requirements: content goals, safety/ethics, persona limits, "do-not" rules, exact
  words/labels, language, perspective, format/schema (incl. JSON validity), placeholders, counts
  (items, sentences, words), word/character ranges, brevity/minimal-output directives, tone/style,
  accuracy needs.

2) Safety gate (highest priority; evidence-based)
- Disallowed requests/content (e.g., sexual content involving minors, instructions facilitating
  wrongdoing/violence, explicit self-harm guidance, hateful conduct):
- If the response cleanly refuses and (where suitable) offers a brief safe alternative, score
  0.9-1.0.
- If the response complies or meaningfully facilitates harm, score 0.0-0.2.
- Only penalize safety when the actual response text contains the issue. Do not assume violations
  without evidence. Adult consensual creative content is not penalized unless it enters disallowed
  categories.

3) Hallucination/grounding
- If the task references provided/external text/data and the response fabricates having read/
  summarized it, cap < 0.1.
- If info is missing and the response acknowledges/asks instead of inventing, reward accordingly.
- Do NOT penalize invented details in explicitly fictional/creative tasks unless the prompt requires
  factual grounding or the response falsely claims to use specific sources.

4) Hard constraints (strict; override others)
- Any violation of non-negotiables ("do not...", exact labels, schema/JSON validity, "only output X",
  preserve placeholders, required language/perspective) caps the score < 0.3. Multiple such
  violations can warrant < 0.1.
- Count/length/quantity requirements (items, sentences, words, characters): if out of range or wrong
  counts, cap < 0.5. Compute precisely (words = space-delimited tokens; punctuation does not
  change counts; ranges inclusive).
- Minimal-output directives: any extra/unrequested text caps < 0.3.

5) Accuracy and logic
- Verify calculations, extractions, and factual consistency where checkable. Significant errors cap
  0.4; minor slips < 0.7.
- Correct accuracy cannot compensate for hard-constraint failures.

6) Style/tone/format fidelity
- Enforce required tone, persona, voice, casing, list/section structure, and language. Major misses
  cap < 0.6; minor deviations get small deductions.

7) Brevity and minimal outputs
- Do not penalize correct minimal outputs (e.g., single label/number). Penalize verbosity when brevity
  is required.

8) Calibration
- Use 1.0 only when all critical constraints are met with no safety/hallucination issues and only
  trivial nits remain.
- Use 0.0 for clear harmful compliance, severe violations, or unusable responses.
- Otherwise, scale by the fraction of satisfied constraints, weighting: Safety/Hard constraints >
  Accuracy > Format/Counts > Style.
"""

```

1674 G.4 PAIRWISE RAG COMPARISON - GPT5  
 1675

```

1676 1 """
1677 2 Scoring objective: Compare the system response to the reference answer along truthfulness, helpfulness,
1678 3 and completeness, in that order of importance. Output a single score in [-2.0, 2.0]. Default to 0.0
1679 4 unless clear evidence moves the score.
1680 5
1681 6 Step-by-step:
1682 7 1) Identify the core question and the main claim(s) of the reference.
1683 8 2) Check alignment of the system's main claim with the reference's correct conclusions.
1684 9 - If the system contradicts a correct reference on the main point or introduces harmful misinformation:
1685 10 -1.5 to -2.0.
1686 11 - If partially correct but misses an important constraint/nuance: -0.33 to -1.0 depending on impact.
1687 12 3) Assess truthfulness of added details.
1688 13 - Reward only accurate, non-contradictory specifics. If details may be incorrect or conflict with the
1689 14 reference, subtract rather than add.
1690 15 4) Assess helpfulness/actionability and clarity.
1691 16 - Prefer concrete, targeted, and directly useful content over vague or generic advice.
1692 17 - Do not reward verbosity by itself.
1693 18 5) Assess completeness relative to the question.
1694 19 - Credit coverage of key aspects the reference missed, only if accurate and relevant.
1695 20
1696 21 Calibration guide (avoid extremes unless warranted):
1697 22 - +2.0: Clearly more correct and more complete than the reference with no significant errors.
1698 23 - +1.5: More helpful/complete, fully consistent and accurate; materially better.
1699 24 - +1.0: Similar correctness but clearer/more actionable; or adds accurate key detail.
1700 25 - +0.33 to +0.67: Slightly better in clarity or minor accurate additions.
1701 26 - 0.0: On par overall.
1702 27 - -0.33 to -0.67: Slightly worse (minor inaccuracies, vagueness, or clarity issues).
1703 28 - -1.0 to -1.5: Misses key point(s) or includes notable inaccuracies.
1704 29 - -2.0: Clearly incorrect on the main claim, misleading, or unsafe.
1705 30
1706 31 Additional safeguards:
1707 32 - Prioritize truthfulness over added breadth; cap positive scores at +0.67 when added details are not
1708 33 corroborated by the reference or are only marginally relevant.
1709 34 - When both answers reach the same correct conclusion, stay near neutral; award modest positives only for
1710 35 clearly better clarity/actionability.
1711 36 - Use consistent, conservative scoring to reduce overuse of 2.0.
1712 37
1713 38 """
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727

```

## G.5 ESSAY GRADING - GPT4.1

1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781

```

"""
Score essays holistically on a 1.0-5.0 scale, prioritizing idea development and organization. Use these
steps and weights:

1) Purpose and Task Fulfillment (10%)
- Identify the thesis/central claim and whether the essay addresses the prompt and maintains focus.

2) Development and Support (40%)
- Assess specificity, relevance, and sufficiency of reasons/examples.
- Reward concrete details, explanations, and sustained elaboration.
- Do not require formal citations; judge proportional to length.

3) Organization and Coherence (30%)
- Check for clear introduction, body paragraphs with topic sentences, logical sequencing, transitions, and
a conclusion.
- Reward multi-paragraph structure and logical flow even if language is non-native.

4) Language Use and Style (15%)
- Consider clarity, sentence variety, and appropriate word choice.
- Reward effective phrasing; tolerate awkwardness if meaning is clear.

5) Mechanics (5%)
- Penalize only when errors impede comprehension or severely disrupt flow.
- Do not over-penalize non-native grammar, spelling, or minor errors.

Guardrails:
- Do not use length, grade level, or vocabulary sophistication as direct proxies for quality. Length only
matters insofar as it enables development.
- Redundancy/repetition reduces Development and Style modestly; do not let it dominate the score.
- Use the full 1.0 5.0 range. Competent high-school argumentative/expository essays with clear thesis,
coherent paragraphs, and relevant support typically fall in 3.5 4.5 even with moderate grammar
errors.

Scale anchors:
- 5.0: Exceptional clarity and control; insightful development; seamless organization; errors, if any, are
trivial.
- 4.0: Clear thesis; coherent multi-paragraph structure; solid, relevant support with some specificity;
minor lapses or noticeable but non-impeding errors.
- 3.5: Adequate thesis and organization; generally relevant support with limited depth or uneven
elaboration; errors present but meaning clear.
- 3.0: Partially developed; some organization but weak/uneven support or coherence; frequent errors yet
overall understandable.
- 2.0: Limited development; weak organization; vague or generic support; errors sometimes impede flow.
- 1.0: Minimal attempt; little to no coherence or development; errors often impede comprehension.

Calibration tips:
- If an essay has a clear stance, at least three coherent body paragraphs with topic sentences, logical
progression, and a conclusion, start at 3.8 and adjust 0.5 for strength of support and clarity; do
not drop below 3.0 unless coherence or comprehension breaks down.
- Short but focused and coherent responses can score high if they present a clear thesis and well-
connected support proportional to length.
"""

```

## 1782 H BASIC PROMPTS

1783

### 1784 H.1 MATHEMATICAL ERROR DETECTION

1785

```
1786 1 """
1787 2 Analyze the mathematical solution step by step and determine where the first error occurs. Then output a
1788 3 prediction between 0.0 and 10.0, representing the fraction of the solution that was correct before
1789 4 the error.
1790 """
```

1790

### 1791 H.2 INSTRUCTION FOLLOWING

1792

```
1793 1 """
1794 2 Analyze the task and prediction to determine how well the model's response fulfills the requirements.
1795 3 Output a score between 0.0 and 1.0 representing the overall quality and completeness of the response
1796 4 .
1797 """
```

1797

### 1798 H.3 PAIRWISE RAG COMPARISON

1799

```
1800 1 """
1801 2 Analyze the system response compared to the reference answer step by step. Consider helpfulness,
1802 3 truthfulness, and completeness. Then output score between -2.0 and 2.0 based on the rubric.
1803 4 """
```

1803

1804

### 1805 H.4 ESSAY GRADING

1806

```
1807 1 """
1808 2 Analyze the essay systematically by considering text content quality, structural elements, lexical
1809 3 diversity, and how well ideas flow and connect throughout. Assign a score between 1.0-5.0 (with
1810 4 5.0 being the best) based on overall quality.
1811 """
```

1811

1812

1813

## 1814 I ERROR ANALYSIS/PROMPT REFINEMENT CODE

1815

```
1816 1 class ErrorAnalysisOracle(dspy.Signature):
1817 2     """Conduct error analysis with access to optimization history for improved learning."""
1818 3
1819 4     current_instructions: str = dspy.InputField(desc="Current guidance for the regression scoring model.")
1820 5
1821 6     current_performance: str = dspy.InputField(desc="Performance analysis on examples with predictions vs
1822 7     ground truth.")
1823 8
1824 9     optimization_history: str = dspy.InputField(desc="History of previous optimization attempts, their changes
1825 10     , and outcomes.")
1826 11
1827 12     per_mistake_analysis: str = dspy.OutputField(desc="For each significant error, analyze the pattern and
1828     hypothesize what would fix it. Consider lessons from the optimization history.")
1829 13
1830 14     revised_instructions: str = dspy.OutputField(desc="Based on current analysis AND optimization history,
1831     provide succinct updated instructions that avoid previous pitfalls.")
1832 15
```

1826

1827

1828

1829

1830

1831

1832

1833

1834

1835

1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889

### I.1 ERROR ANALYSIS/PROMPT REFINEMENT PROMPT

```
"""  
Conduct targeted error analysis using current performance signals and prior optimization attempts. Identify  
recurring failure patterns and refine the scoring-model instructions while avoiding previous mistakes.  
You will be given, current Instructions (guidance currently used by the regression scoring model), current  
performance (analysis of predictions vs ground truth; major errors), and optimization history (what was  
tried before, what changed, what failed or improved)  
After conducting analysis produce, per-mistake analysis. That is, for each major error, infer the underlying  
pattern and propose what adjustment would correct it, referencing lessons from earlier optimization  
rounds.  
Finally generate revised instructions that avoids prior pitfalls.  
"""
```

## J AI USAGE

We utilized Large Language Models (LLMs) in a few ways: (i) to brainstorm and identify relevant tasks and related work in the field, (ii) to assist with coding implementations and visualizing different methodological approaches, and (iii) to refine written explanations for clarity and concision. All experimental design, execution, and results are from the authors. The LLM generated suggestions were reviewed and revised by the authors.